**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**




**Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación**

# TRABAJO FIN DE GRADO

## ESTUDIO DE CICLO LIMITE EN CONTROL DIGITAL DE CONVERTIDORES CONMUTADOS Y TECNICAS DIGITALES PARA EVITARLO


**Juan Álvaro Mora Fraile**
**Tutor: Ángel de Castro Martín**


**JULIO 2019**

# ESTUDIO DE CICLO LIMITE EN CONTROL DIGITAL DE CONVERTIDORES CONMUTADOS Y TECNICAS DIGITALES PARA EVITARLO

**Juan Álvaro Mora Fraile**
**TUTOR: Ángel de Castro Martín**

**HCTLab**
**Dpto. Tecnología Electrónica y de las Comunicaciones**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Julio de 2019**

# Resumen (castellano)

Este Trabajo Fin de Grado tratará sobre el estudio de las oscilaciones indeseadas en sistemas de lazo cerrado controlados mediante tecnología digital llamadas ciclo límite, desglosando su origen, en qué condiciones aparece, los problemas que conlleva y una técnica para eliminarlo o atenuarlo. Al diseñar un sistema digital con una salida estable, pueden aparecer oscilaciones indeseadas de carácter aleatorio y de frecuencias muy bajas, lo que dificulta su eliminación.

La oscilación del ciclo límite aparece solamente en reguladores digitales y la condición para que se dé es que estos cuenten con una resolución de ADC mayor que la del módulo PWM. De esta forma, se podrán leer valores en la tensión de salida con mayor precisión que la que tienen a la hora de actuar sobre la propia salida. La consecuencia directa de este hecho es la aparición de oscilaciones en el error, es decir, variará entre valores cercanos a cero sin llegar a serlo, y, con ello, la actuación sobre el sistema será variable y oscilante.

La solución que propone este Trabajo Fin de Grado es la que se presenta en el artículo de Angel V. Peterchev y Seth R. Sanders llamado "Quantization Resolution and Limit Cycling in Digitally Controlled PWM Converters", en el cual se convierten esas oscilaciones no controladas y poco predecibles en unas de mayor frecuencia y menor amplitud para su posterior eliminación mediante un filtro. Esta técnica es denominada "Dither" (del inglés "temblor"), ya que su fundamento es variar la actuación del módulo PWM sobre dos valores cercanos de manera periódica, de forma que la salida no sea fija y, con ello, reducir la oscilación indeseada de salida.

En este Trabajo Fin de Grado se verán ejemplos prácticos de situaciones en las que aparece el ciclo límite y cómo se elimina o reduce tras aplicar la técnica.

# Abstract (English)

This Bachelor Thesis will deal with the study of unwanted oscillations in closed-loop systems controlled by digital technology called limit cycling, breaking down its origin, in which terms it appears, problems that entails and a technique to erase or reduce its effects. Designing a digital system with a stable output, random unwanted oscillations of very low frequencies can emerge, which can be difficult to remove.

The limit cycling oscillation only appears in digital regulators and the condition for this appearance is that ADC resolution is higher than the PWM module. Thus, they can read in the output voltage with greater precision than acting on the output. The direct consequence is the apparition of oscillations in the error that will vary between values close to zero without becoming zero and the action on the system will be variable and oscillating.

The solution proposed by this Bachelor Thesis is the one presented in the article by Angel V. Sanders and Seth R. Sanders called "Quantization Resolution and Limit Cycling in Digitally Controlled PWM Converters". In this paper, these uncontrolled and unpredictable oscillations are converted into ones of higher frequency and lower amplitude for later elimination using a filter. This technique is called "Dither", and it is based in the variation of the actuation of the PWM module on two nearby values periodically and, thus, reduce the unwanted oscillation in the output.

Practical examples of situations in which the limit cycling appears are seen and how it is eliminated or reduced after applying the technique.

# Palabras clave (castellano)

Oscilación indeseada, ciclo límite, regulador, sistemas digitales, convertidor analógico-digital, modulador de ancho de pulso, teoría de control, Dither, actuación

# Keywords (inglés)

Unwanted oscillations, limit cycling, regulator, digital systems, analog-to-digital converter, pulse width modulation, control theory, Dither, actuation

*Agradecimientos*

# INDICE DE CONTENIDOS



# INDICE DE FIGURAS

Y

# INDICE DE TABLAS

# 1 Introducción

## 1.1 Motivación

Esta memoria de TFG surge gracias al artículo de Angel V. Peterchev y Seth R. Sanders llamado "Quantization Resolution and Limit Cycling in Digitally Controlled PWM Converters", publicado en enero de 2003, el cual trata sobre en qué condiciones aparece la oscilación del ciclo límite, que es una oscilación de baja frecuencia no deseada que aparece en sistemas en lazo cerrado con regulador digital, y cómo se elimina. Plantea como solución la implementación de la técnica del Dither, y este Trabajo se basa en este artículo a la hora de implementarlo.

## 1.2 Objetivos

El objetivo de este Trabajo es el estudio en profundidad del ciclo límite, sea su origen, en qué condiciones aparece y cómo paliarlo o, incluso, eliminarlo. Para ello, es necesario tener conocimientos de teoría de control, diseño de reguladores, codificación en VHDL y electrónica avanzada.

## 1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Introducción a sistemas electrónicos y teoría de control: se hablará de los elementos que compondrán el sistema de prueba, así como de una breve explicación sobre ecuaciones en diferencias y su aplicación en control digital.
- El ciclo límite, su origen y problemática
- La técnica del Dither: se describirá el método que nos propone el artículo "Quantization Resolution and Limit Cycling in Digitally Controlled PWM Converters" [Peterchev03].
- Presentación del caso de estudio y los componentes que conformarán el sistema: se hablará sobre las plantas (filtro RC y convertidor Buck) sobre las que aplicaremos el sistema de prueba.
- Diseño de los bloques: describirá el funcionamiento de cada bloque del sistema final (gestorADC, moduloPWM, regulador y controlador).
- Desarrollo del trabajo: tratará sobre la implementación de cada bloque y su adición al sistema final de prueba.
- Resultados: hablará sobre la obtención de las medidas mediante simulaciones y experimentos reales con la ayuda gráfica de figuras y tablas.

# 2 Estado del arte

## 2.1 Descripción del sistema de trabajo

Un controlador es un dispositivo con una o varias entradas analógicas y una o varias salidas digitales, cuyo objetivo es gobernar la actuación de un sistema o planta de forma que esta sea estable, precisa o que tenga una forma de onda concreta (en nuestro caso será constante). A continuación, se mostrará un esquema del modelo de controlador que utilizaremos en este Trabajo.

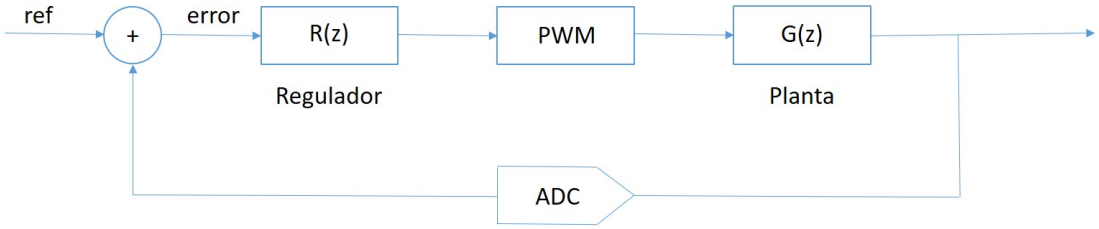


Figura 2-1. Modelo del controlador

Al ser un sistema en lazo cerrado, vamos a explicar el funcionamiento del controlador bloque a bloque de izquierda a derecha. El primer bloque que se tiene es un restador cuyas entradas son la consigna deseada y la lectura del convertidor analógico-digital (a partir de ahora lo se llamará ADC). La diferencia es el error. A la salida del restador tenemos el regulador, que con la ayuda de una ecuación en diferencias que se ha diseñado, tiene la función de generar el valor digital del ciclo de trabajo para que el bloque del modulador de ancho de pulso (a partir de ahora lo se llamará PWM) excite a la planta para que tenga la forma de onda deseada. Siguiendo el orden, nos situamos en el módulo PWM, que generará una señal digital que servirá como analógica para controlar la salida del sistema. La planta es el subsistema que queremos gobernar. En este Trabajo se utilizarán dos sistemas en el estudio: un filtro RC y un convertidor conmutado de tipo Buck. La salida de la planta es la del sistema total, e irá conectada a un ADC, cuyo objetivo es convertir la señal analógica de la salida a un conjunto de bits que se conectarán al restador inicial. Es muy importante que el ADC muestree en el mismo momento de cada ciclo de conmutación, es decir, que lo haga de forma periódica para evitar que leamos la tensión en distintos puntos del rizado de conmutación y, con ello, aumentar el error en la actuación.

### 2.1.1 Reguladores digitales

El regulador es el corazón del sistema. Es el elemento digital que se encarga de controlar la tensión de salida a partir de una entrada concreta y un valor de referencia o consigna con la ayuda de una ecuación en diferencias, definida por el diseñador [DeCastro03]. Hay que tener en cuenta que esta ecuación depende no solo de parámetros del controlador como la frecuencia de conmutación, sino que es importante saber cómo se define la planta que vamos a controlar, aunque se hablará de ellas más adelante.

Las ecuaciones diferenciales se pueden describir como una división de polinomios definidos en el plano z mediante la transformada Z, por lo que se puede hablar de ceros y polos. Un cero es un valor de z que hace que la ecuación valga cero y un polo es uno que hace que la igualdad tienda a infinito. El valor de estos ceros y polos modelarán la

respuesta del regulador, haciéndola más rápida, más lenta, con sobreoscilación, con un error concreto o con otros comportamientos.

Es imprescindible que, en el sistema total, es decir, contando con los ceros y polos de la planta, el número de ceros sea menor o igual al de los polos, ya que un cero supone un adelanto en el dominio temporal, y es imposible que un sistema causal pueda obtener valores exactos y futuros en un instante concreto.

Dentro de los reguladores lineales, se pueden distinguir según su disposición de ceros y polos:

- Reguladores tipo K o proporcional: simplemente escala el valor de entrada.

$$R(z)=k$$

- Reguladores PD o proporcional derivativo: no hace el error nulo porque no resta la muestra anterior escalada por 'a' con la actual, siendo 'a' menor que 1.

$$R(z)=\frac{k*z-a}{z}$$

- Reguladores PI o proporcional integrativo: consigue el error nulo al comparar la muestra anterior con la actual, siendo 'b' menor que 1.

$$R(z)=\frac{k*z-b}{z-1}$$

- Reguladores PID o proporcional integral-derivativo: consigue también el error nulo gracias al polo integrador o (z – 1). Es una mezcla del tipo PI y PD.

$$R(z)=\frac{k*(z-b)*(z-a)}{z*(z-1)}$$

Dentro de los requisitos que se pueden imponer a la hora de diseñar un regulador, es destacable el tiempo de establecimiento, que es el tiempo que tarda la salida a estar al 95% del valor máximo al que debe llegar, es decir, si el valor deseado es de 3V, el tiempo de establecimiento será el tiempo que tarda desde los 0V hasta 2,85V; y la sobreoscilación, que es el porcentaje de valor máximo que alcanza la señal de salida siendo este superior al valor que le hemos pedido, es decir, si se quiere que el regulador nos devuelva 5V, será sobreoscilación todo valor superior a 5V sin contar con el rizado de conmutación.

### 2.1.2 Elementos de procesamiento de señal

Estos componentes son los encargados de hacer todas las tareas necesarias para el correcto funcionamiento del controlador: tomar muestras del ADC periódicamente, realizar las operaciones del regulador y actuar sobre la planta mediante un PWM. Para este trabajo se dispone una amplia gama de subsistemas como los microprocesadores, los DSP y las FPGA.

Los microprocesadores son circuitos integrados con la capacidad de hacer operaciones mediante instrucciones codificadas [DelBrio99][DeCastro03]. Estos pueden funcionar a frecuencias de reloj superiores al GHz, pero para hacer una sola operación necesitan varios ciclos.

Los DSP son procesadores especializados en el procesamiento de señal. Se diferencian de los anteriores en que estos tienen un repertorio de instrucciones mucho más reducidos, pero más optimizado[DeCastro03]. Nuevamente, pueden ejecutar instrucciones a frecuencias muy grandes, pero tienen el inconveniente de que necesitan varios ciclos por instrucción.

Por último, las FPGA son dispositivos formados por un conjunto de bloques aritméticos, multiplexores, bloques de memoria, registros y de muchos otros tipos. La ventaja de las FPGA es que podemos programar el hardware, convirtiéndola en un sistema hecho exprofeso para una sola aplicación [Sutter06] [DeCastro03] [XilinxFPGA]. La gran ventaja es su capacidad de paralelización de las tareas, por ejemplo, de la actuación del PWM, lo que hace que sean más rápidas que los microprocesadores y los DSP pese a que funcionan a frecuencias mucho menores (centenas de MHz), a costa de utilizar gran cantidad de recursos electrónicos (memoria, sumadores, …).

### 2.1.3 Plantas a controlar

Una planta es el sistema que se quiere controlar y su comportamiento está definido por ecuaciones en diferencias, las que se utilizarán para diseñar un regulador que cumpla los parámetros que se han comentado anteriormente.

En este Trabajo, las plantas que se emplearán son un filtro RC paso bajo y un convertidor conmutado tipo Buck.

Un filtro RC es un filtro pasivo cuya función es atenuar las frecuencias no deseadas de la señal de entrada y dejar pasar a las que se quieren [Malik95]. En nuestro caso, al ser paso bajo, no se eliminarán las frecuencias menores a la de diseño, y al tener una resistencia de 4,7 kΩ y un condensador de 22nF la frecuencia de corte es 1,539 kHz. La siguiente figura mostrará el esquema circuital que sigue el filtro en cuestión.

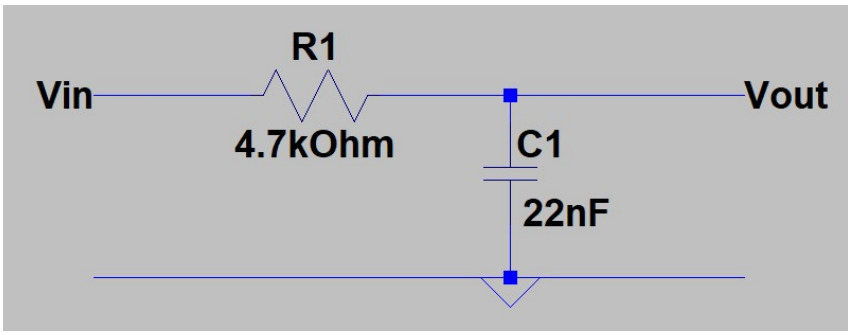


Figura 2-2. Esquema del filtro RC

Un convertidor conmutado tipo Buck [Malik95] [Erickson01] de continua-continua es una fuente de alimentación que puede generar una salida de tantos voltios como se quieran, limitado por la tensión de la entrada, es decir, si la entrada tiene 12V no se pueden sacar 15V, pero sí valores menores a 12V, ya que la eficiencia de estos convertidores no es de

100% debido a perdidas en los MOSFET de paso y las resistencias parásitas de la bobina y el condensador [Malik95]. Cabe destacar que a la salida se le asocia un rizado producido al activar y desactivar el MOSFET de paso, por lo que la tensión crecerá y decaerá a lo largo de un ciclo de conmutación, generando un rizado llamado rizado de conmutación.

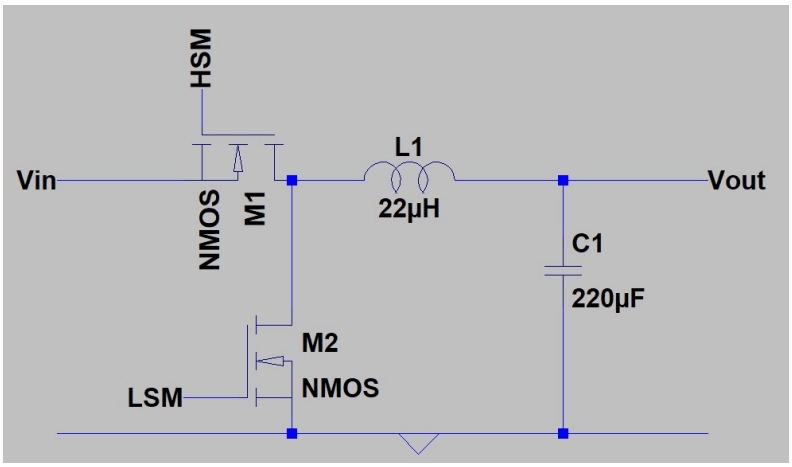


Figura 2-3. Esquema del convertidor Buck

## *2.2 Oscilaciones indeseadas: el ciclo límite*

El ciclo límite [Peterchev03] es un efecto de oscilación indeseada que aparece en la salida solo en reguladores digitales a pesar de que estos sean estables. Este fenómeno surge debido a que el número de bits de lectura del ADC es mayor al de los de actuación del PWM, es decir, cuando la precisión de la actuación es menor que la precisión de la medida, de tal forma que resulta imposible llegar a un valor de régimen permanente en la salida que anule el error. En la siguiente figura explica gráficamente la aparición de la oscilación del ciclo límite o LCO.

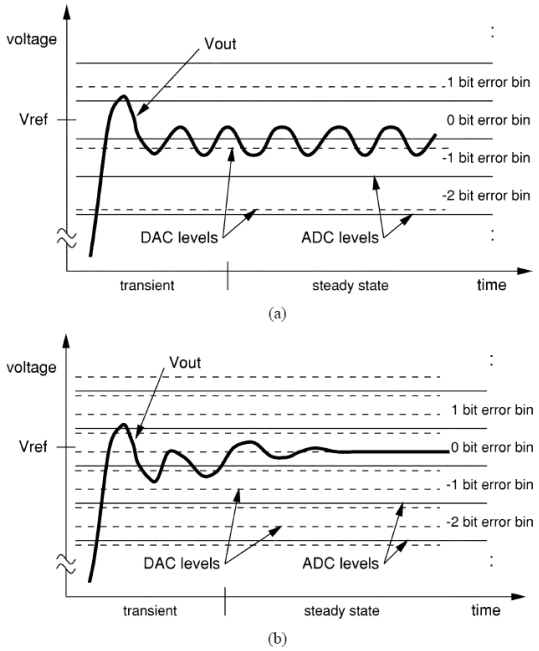


Figura. 2-4. Forma de onda de la salida con LCO (a) y sin LCO (b) [Peterchev03]

### 2.2.1 Origen del ciclo límite

A parte de los elementos que ya se han nombrado, los controladores también poseen un convertidor analógico-digital y un módulo PWM.

El convertidor analógico-digital o ADC es un componente que permite obtener una lectura digital sobre una tensión analógica [DeCastro03]. Para ello, utiliza una tensión de referencia, que será el valor de lectura, y lo compara con la entrada que se quiere traducir a un número concreto de bits que tienen de serie.

El módulo PWM, del inglés "Pulse Width Modulation", es un dispositivo que genera una señal digital, de la que se puede modificar su ancho en un rango definido. Consiste en un contador que, cuando llega a un valor concreto o consigna, activa la señal de salida a "0"; mientras tanto, esta señal se mantiene a "1". La consigna tiene un número contable de bits, por lo que los posibles anchos de señal se ven limitados a la potencia en base 2 de dicho número.

Echando un vistazo al modelo del controlador, se puede observar que la salida del sistema depende del ciclo de trabajo (en nuestro caso es el ancho de la señal PWM), que a su vez depende directamente del error que comete el regulador, el cual vuelve a depender de la lectura del ADC, por lo que el módulo PWM está relacionado con la lectura del ADC. La cuestión es que, si el controlador es capaz de actuar con más precisión que la lectura del ADC, tendremos una salida del sistema estable. Por ejemplo, si el ADC tiene 8 bits y el PWM 10 bits. En caso contrario, se podría tener ciclo límite porque será, en algunas consignas, imposible reducir el error a cero y la salida será de carácter oscilante.

### 2.2.2 Problemas que conlleva

El ciclo límite lleva asociada una señal acoplada a la de salida con una frecuencia más baja que el ciclo de conmutación y no predecible, por lo que sería muy difícil conseguir filtrar dicha frecuencia ya que los filtros paso bajo no son suficiente para eliminar la oscilación. Además, la señal del LCO puede tener una amplitud bastante grande. Estos sucesos van en contra de la lógica del diseño de un regulador lineal, ya que el objetivo es que la salida del sistema sea constante o prácticamente constante, salvo por el rizado de conmutación.

### 2.2.3 Soluciones: el Dither

Tras indagar en el origen del ciclo límite, la primera solución, y la más simple, sería aumentar el número de bits de actuación del PWM hasta que este supere a los de lectura del ADC. El problema es que al subir en bits el PWM, se debe aumentar la frecuencia de reloj del sistema para que la frecuencia de conmutación sea igual a la original, ya que esta viene impuesta por el diseño del convertidor que se quiere controlar.

Otra posible solución puede ser bajar el número de bits de lectura del ADC hasta que este sea menor al número de bits del PWM. Nuevamente, surgiría un problema: la precisión de lectura del ADC sería muy baja y, por tanto, influye en la precisión del regulador, el cual creería que el error es nulo, pero la banda de error cero sería grande y podría tener un error real nada despreciable cuando el regulador cree que está con solución buena por la baja resolución de medida del ADC y, con ello, no cumplir las especificaciones impuestas.

La solución que se va a aplicar en este Trabajo es el Dither. El Dither [Peterchev03] consiste en variar el ciclo de trabajo con el objetivo de estabilizar la oscilación de salida. La forma de onda de la salida del controlador, en este caso, debe ser constante (sin contar con el rizado de conmutación), pero se sabe que, en ciertas condiciones, concretamente cuando el número de bits del ADC es mayor al de los del PWM, aparecerán oscilaciones de bajas frecuencias indeseadas, llamadas “ciclo límite”. Por ejemplo, si se tiene un resto constante de 5, en cada ciclo de conmutación el acumulador sumará “5” al valor anterior que tenía hasta llegar a “16” o más, donde restará “16” y continuará con el resultado (5, 10, 15, 21→6, 11, 16→0, …). La siguiente figura ayudará a entender el proceso del Dither:

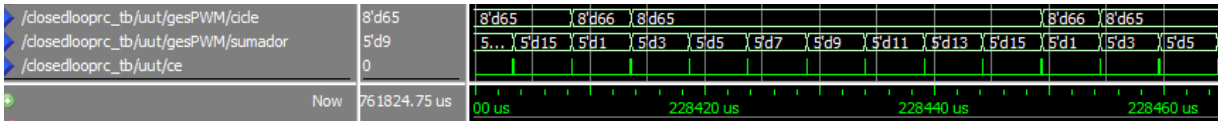


Figura 2-5. Funcionamiento del Dither

La ventaja de esta solución es utilizar esta condición para intentar conseguir una mínima oscilación y de frecuencia conocida y mayor que la del ciclo límite, lo que permite filtrarla posteriormente.

En este Trabajo, en el peor caso de LCO corresponde con una lectura de ADC de 12 bits y una actuación de PWM de 8 bits. El Dither es una versión mejorada del módulo PWM y se implementa de la siguiente forma: la salida del regulador, de 12 bits, se conecta al módulo PWM con la mejora, de forma que se utilizarán los 8 más significativos para el ciclo de trabajo y los 4 de menor peso (a partir de ahora se llamarán “resto”) irán asociados a un acumulador de tantos bits como tenga el resto más “1” para evitar desbordamientos en la suma. En cada ciclo de conmutación, este acumulador sumará el resto del ciclo actual. Si

dicho el bit más significativo del acumulador es “1”, sumaremos “1” al ciclo de trabajo y pondremos ese bit a “0”, de forma que el ancho de pulso de salida variará para compensar la futura oscilación.

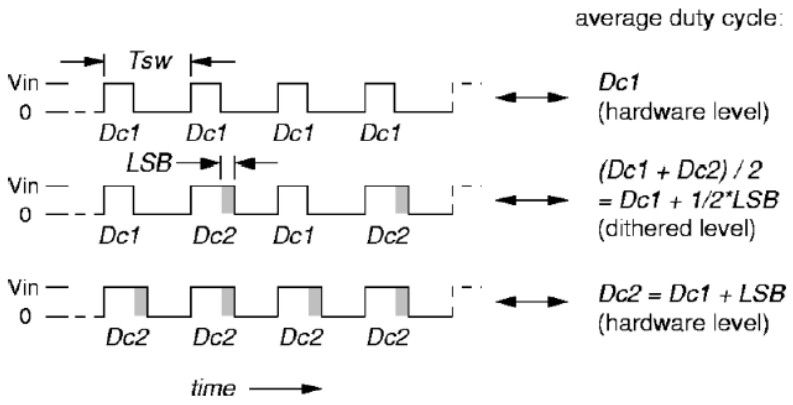


Figura 2-6. Forma de onda del PWM con Dither [Peterchev03]

Como norma general, si el número de bits del ADC es menor o igual que el del PWM sumados con los del resto, el efecto del ciclo límite se atenúa mucho comparado con el caso sin la solución, o incluso puede eliminarse por completo. Este método es el que se ha utilizado en este Trabajo y su implementación se explicará en el apartado 4 y los resultados obtenidos en el 5.

# 3 Diseño

## 3.1 FPGA

Se ha utilizado el modelo Spartan 3 XC3S200FT256 de Xilinx, lo que significa que tiene 200.000 puertas lógicas y el encapsulado que se le ha aplicado es de 256 pines [XilinxFPGA]. Este chip se encuentra alojado en una tarjeta con diversos periféricos, como un puerto VGA, uno RS-232 o varios pines de entrada/salida, llamada Spartan-3 Starter Kit Board [XilinxBoard].

A parte de tener 12 multiplicadores hardware de 18 por 18 bits, este integrado cuenta con un oscilador de cristal de 50 MHz, frecuencia que podremos aumentar o disminuir a voluntad gracias a sus 4 DCM (digital clock manager) [XilinxFPGA]. La tensión máxima de entrada o salida de los pines externos es de 3,3V, por lo que una tensión mayor podría dañarlos, o incluso romperlos.

La frecuencia que utilizará el diseño es la frecuencia base de la FPGA, que es de 50 MHz. Por tanto, como el contador del módulo PWM tiene 8 bits, la frecuencia de trabajo, también llamada de conmutación, será de 195,3125 kHz.

## 3.2 Convertidor analógico-digital (ADC)

Para poder estudiar el fenómeno del ciclo límite, se necesita un ADC que tenga muchos más bits de resolución que el módulo PWM, que tiene 8. Por ello, se ha optado por el modelo LTC1273 de Linear Technology, de 12 bits de resolución en encapsulado DIP-24 [LinearADC], para facilitar su integración en una placa protoboard.

Este componente se caracteriza por su capacidad para realizar 300 kSPS (muestras por segundo). Es necesario que pueda aportar un número mayor de muestras que la frecuencia de conmutación porque, a la hora de calcular el error, el controlador necesita una lectura de tensión traducida al mundo digital cada ciclo de conmutación.

Otro dato importante de este ADC traduce valores de tensión entre 0 y 5V. Esto supondrá un problema a la hora de implementar el modelo del Buck, el cual aporta hasta 12V de voltaje.

Para terminar con las características, la entrada analógica de este integrado tiene una impedancia muy alta, lo que significa que se podrá conectar directamente la salida del modelo al pin del chip, y las conversiones serán de alta fidelidad gracias a la SNR tan alta que posee (70 dB en 150 kSPS) [LinearADC].

A la hora de pedir una lectura de tensión desde nuestra FPGA, el LTC1273 utiliza un protocolo de comunicaciones bastante sencillo. Para explicar su funcionamiento, a continuación, se mostrará una imagen con las formas de onda que sigue según la hoja de datos del fabricante.

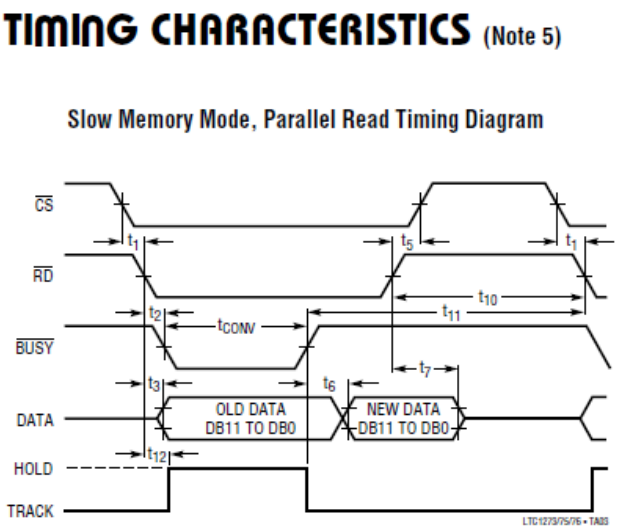


Figura 3-1. Procotolo del ADC LTC1273

Dentro de todas las modalidades de funcionamiento que dispone este ADC, se ha implementado por el "Slow Memory Mode, Parallel Read" porque en una misma petición devuelve los 12 bits de conversión a la vez y también permite muestrear a la frecuencia de conmutación.

## *3.3 Regulador digital*

Una vez se tiene todo el hardware seleccionado y se conoce el comportamiento de las plantas que se van a controlar, es decir, se sabe su ecuación en diferencias, se puede empezar a pensar de qué tipo debería ser el regulador. Al ser un Trabajo de investigación, las especificaciones no son muy limitantes: error nulo y la posibilidad de que sobreoscile ligeramente. Por último, hay que tener en cuenta la frecuencia de conmutación, ya que esta hará que el tiempo de establecimiento del regulador varíe para una misma ecuación del propio regulador.

Para el diseño se ha utilizado un código en Matlab, el cual describe la planta objetivo y ejecuta una aplicación específica para esta tarea mediante el comando "sisotool(Gz)", donde "Gz" es la ecuación en diferencias de la planta.

El proceso utilizado para hallar el regulador es común a todas las plantas: con Matlab y el plano z de la planta, se añaden ceros y polos para modelar el comportamiento deseado y se modifica la ganancia para ajustar la sobreoscilación. Como las especificaciones no hablan sobre tiempos de establecimiento mínimos, se puede bajar la ganancia tanto como sea necesario para mitigar la sobreoscilación. En cuanto al error nulo, se debe añadir obligatoriamente un integrador en el regulador.

### 3.3.1 Regulador del filtro RC

La ecuación en diferencias resultante del regulador es $\frac{0,001}{z-1}$. Gracias al integrador se obtemdrá un error nulo, la ganancia tan pequeña no dará ninguna sobreoscilación y su

tiempo de establecimiento es de 12,8 ms.

La aplicación de Matlab muestra el siguiente comportamiento:

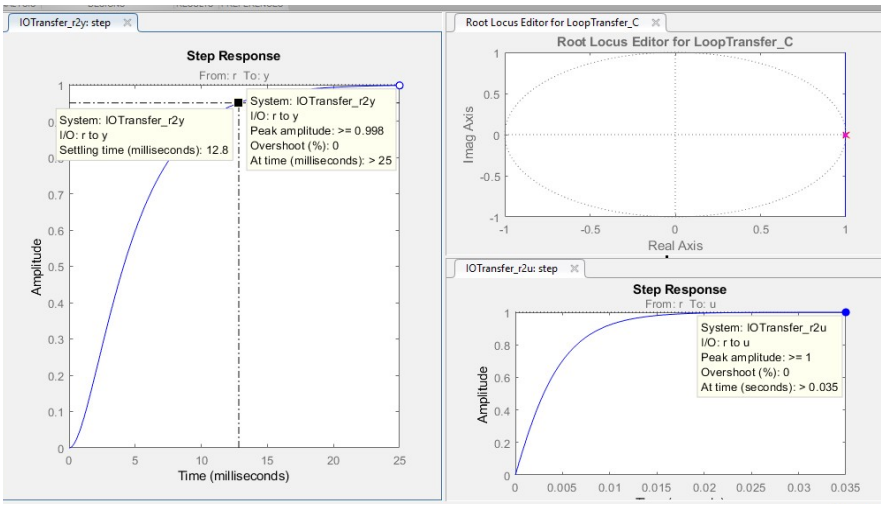


Figura 3-2. Tiempo de establecimiento del regulador del filtro RC

### 3.3.2 Regulador del convertidor Buck

El diseño final del regulador del Buck tuvo el mismo procedimiento que los de los filtros RC. En este caso, la ecuación en diferencias final es $\frac{0,083*(z-0,96)*(z-0,97)}{z*(z-1)}$. Nuevamente, el integrador asegura que el error sea cero, la escasa ganancia evita la sobreoscilación, los ceros hacen que el tiempo de establecimiento sea pequeño y el polo en cero evita que el sistema sea inestable.

La siguiente figura mostrará el comportamiento del sistema:

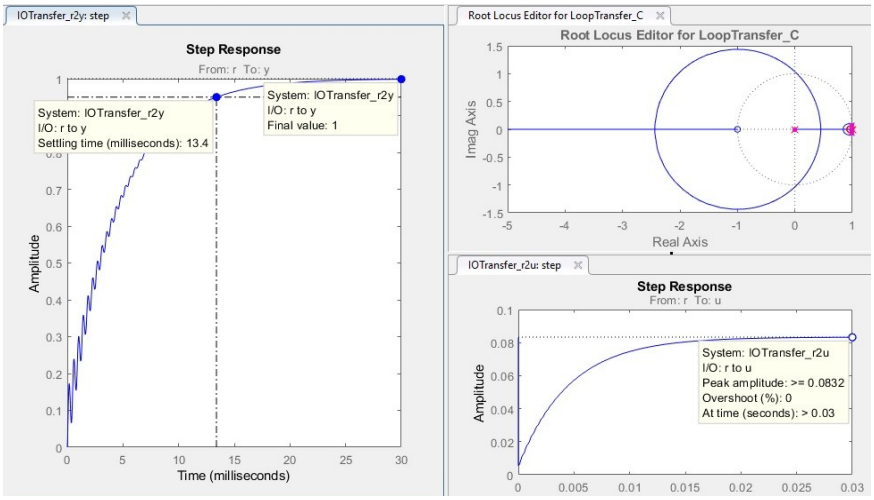


Figura 3-3. Tiempo de establecimiento del regulador del convertidor Buck

Podemos destacar que en la cresta de subida que tiene la actuación hay una forma de onda aserrada, que se debe al efecto que tiene la planta sobre todo el conjunto.

# 4 Desarrollo

A la hora de realizar este Trabajo, se ha partido del controlador utilizado en las prácticas de la asignatura Sistemas de Control. Este constaba de un ADC de 8 bits y de un PWM de 8 bits de resolución, por lo que no podríamos ver las oscilaciones típicas del ciclo límite, pero sirvió como punto de partida.

## 4.1 El gestor del ADC

Lo primero que se tuvo que hacer era cambiar el ADC. Esto suponía, hablando en VHDL, de sustituir el bus de 8 bits de salida del gestor del ADC por uno de 12 y modificar el protocolo de comunicación con el chip, haciéndolo apropiado para el LTC1273. Tal y como se ha comentado en el apartado 3.2, el modo de funcionamiento que se ha decidido que utilizará el ADC es el "Slow Memory Mode, Parallel Read", el cual dará medidas de 12 bits de resolución. El protocolo de dicha modalidad utiliza tres señales de control: CS, RD, BUSY, siendo todas de activación baja, es decir, su señal de activación es un '0' en vez de un '1'. La figura 3-1 muestra el proceso gráficamente desde que la FPGA hace una petición de conversión al ADC hasta que la consigue:

1. Para indicar que se quiere un dato, desde la FPGA se escribe en CS y RD un '0' (las activamos).
2. Tras pasar un tiempo que el fabricante del ADC ha proporcionado (270 ns como máximo), el chip habilita la señal BUSY.
3. Tras activar BUSY, el fabricante vuelve a dar un tiempo máximo de conversión (3 us), BUSY se pone a '1' y tras 100 ns, la conversión de la tensión queda resuelta: el ADC devuelve su valor en binario a través del bus de 12 bits del que dispone.
4. Para finalizar la conversión, se debe desactivar las señales CS y RD mediante la FPGA.

A la hora de implementar el protocolo en VHDL, se utilizó un contador de 8 bits, lo que define el ciclo de conmutación. Ese contador será utilizado también a modo de cronómetro para poder procesar el protocolo respetando los requisitos del fabricante y, al final de la cuenta, activará el CE, que marcará el ciclo de conmutación como si de un divisor de frecuencia de tratase.

Para terminar, hay que tener en cuenta que el ADC devuelve valores de entre 0 y 5V, por lo que cada nuevo dato debe ser escalado de forma que el máximo del chip (5V) se convierta al máximo del patillaje de la FPGA (3,3V). Para ello, la conversión se multiplica por 1,515 (que corresponde con 5/3,3) porque como el valor de tensión máximo que devuelve la FPGA por sus pines es 3,3V, las lecturas del ADC serán de 3,3V como máximo, por lo que es necesario convertir ese valor a otra escala.

Las siguientes figuras muestran el código VHDL que modela este proceso.

```
elsif (rising_edge (Clk)) then

    if ((ADC_ready = '1') and (ADC_copy = '1')) then
        ADC_aux <= to_integer(unsigned(ADCin));
        adcfixed <= to_sfixed(ADC_aux, 13, 0);
        --registredADC <= ADCin;
        registredADC <= resize(escalado * adcfixed, registredADC);
        ADCOut_aux <= to_slv(registredADC);
        ADCOut <= ADCOut_aux(18 downto 7);
        ADCregistrado <= ADCOut;
    end if;
```

Figura 4-1. Código del escalado del ADC

```
case contador is

    when x"10" =>
                    nCS <= '0';
                    nRD <= '0';
    --when x"0a" => espera a que la señal BUSY sea '0'

    --when x"90" => termina la conversion
    -- el tiempo desde que pedimos la conversion hasta que la tenemos es de 155 ciclos (9b = 3.1us)
    -- se deja un pequeño margen

    when x"b0" =>   --si el dato esta listo para ser leido
                    ADC_ready <= '1';

    -- pasa 1 us para que se pueda hacer otra conversion

    when x"fe" =>   --activa CE  para que el regulador haga un calculo por ciclo de trabajo
                    CE <= '1';

    when x"ff" =>   -- indicamos que el dato ha sido leido
                    nCS <= '1';
                    nRD <= '1';
                    ADC_ready <= '0';
                    CE <= '0';

    when others => null;
end case;
```

Figura 4-2. Código del protocolo del ADC

## 4.2 El regulador

En el caso del bloque regulador, es imprescindible conocer previamente el dispositivo que se desea controlar, como se ha comentado en el apartado 3.3. Como era de esperar, la ecuación en diferencias que ejecuta el regulador para gobernar al filtro RC no será la misma que el del Buck, aunque sí compartirán el resto de la funcionalidad.

Las entradas y salidas de este bloque se mostrarán con la ayuda de la siguiente figura. Para los casos sin Dither, la actuación tendrá 9 bits (1 de signo y 8 de valor), y para los que tienen Dither, la actuación será de 13 bits (1 de signo y 12 de valor).

```
entity Regulador is
    Port ( Clk : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           CE : in  STD_LOGIC;
           Error : in  STD_LOGIC_VECTOR (12 downto 0);
           Actuacion : out  STD_LOGIC_VECTOR (8 downto 0));
end Regulador;
```

Figura 4-3. Entidad genérica del regulador

A la hora de implementar el regulador, hablando en VHDL, se comienza haciendo que la salida de este sea dependiente de CE, que es la señal que indica al sistema el inicio del ciclo de conmutación, y haciéndolo síncrono, aunque el propio CE lo sea. Después, hay que tener en cuenta que el hardware se maneja en el dominio temporal, por lo que se debe que convertir la transformada Z que controla el comportamiento del regulador en una ecuación que dependa de entradas y salidas. Esto es tarea fácil si se conoce que los polos son retrasos en el tiempo y que los ceros son adelantos. Utilizando esta regla, es posible describir cualquier transformada Z del tipo división de polinomios en una ecuación en la que se suman o restan entradas y salidas escaladas. Todos estos términos se procesan en coma fija, es decir, el número de bits que forman la parte entera y los decimales es constante. A continuación, se mostrará una figura que contiene la implementación de la ecuación en diferencias pasada a dominio temporal. Las variables "Xx" representan las entradas con 'x' retrasos y "Yy" son las salidas con 'y' retrasos.

```
-- Cálculo de las operaciones intermedias
sB0X0 <= resize(sX * B0, sB0X0);
sB1X1 <= resize(sX1 * B1, sB1X1);
sB2X2 <= resize(sX2 * B2, sB2X2);
sA1Y1 <= resize(sY1 * A1, sA1Y1);
sA2Y2 <= resize(sY2 * A2, sA2Y2);

-- Calcular el valor de la salida actual
-- Mantener gran resolución para no perder información
sY_max <= resize (sB0X0 + sB1X1 + sB2X2 - sA1Y1 - sA2Y2, sY_max);

sY_sat <=  MAX_sY when (sY_max > MAX_sY) else
           MIN_sY when (sY_max < MIN_sY) else
           resize (sY_max, sY_sat);

-- Transformar el valor al tamaño de salida
sY <= resize (sY_sat,sY);

end Behavioral;
```

Figura 4-4. Implementación de la ecuación en diferencias

Para terminar, es necesario que los términos "Xx" y "Yy" tengan mucha resolución, es decir, que puedan representar números con muchos decimales, porque como las oscilaciones del ciclo límite van a aparecer cuando el error sea cercano a cero, necesitamos que la actuación sea lo más precisa posible para que el regulador pueda distinguir entre dos valores cercanos.

### 4.2.1 El regulador del filtro RC

Cabe destacar que el comportamiento en la transformada Z que se ha obtenido es $\frac{0,001}{z-1}$, tal y como se ha comentado en el apartado anteriormente. Pero lo más relevante es la necesidad de escalar este resultado por un factor que dependerá de la diferencia de bits entre la entrada del regulador (siempre 12) y la salida, que, a su vez, dependerá de si el módulo PWM tendrá Dither (la salida será de 12 bits) o no (8 bits). Dicho esto, siempre que no haya Dither, la ecuación en diferencias en el dominio temporal se escalará por 0,0625, que corresponde con $2^{-4}$. En cambio, si el PWM tiene Dither, la ecuación queda como está.

En VHDL, el regulador de los casos sin Dither para el filtro RC tendrán la siguiente declaración de términos de la ecuación, representada en la siguiente figura.

```
--entrada y salida
signal sX : sfixed (12 downto 0);
signal sY : sfixed (8 downto 0);

--valores anteriores de la entrada
signal sX1 : sfixed (12 downto 0);
signal sX2 : sfixed (12 downto 0);

-- coeficientes
constant B0 : sfixed (2 downto -15) := to_sfixed(0.0625*0.001 ,2 ,-15 );            --1/16
constant B1 : sfixed (2 downto -15) := to_sfixed(0 ,2 ,-15 );       --(-0.99)/16
constant B2 : sfixed (0 downto 0) := to_sfixed(0 ,0 ,0 );                  --0/16
constant A1 : sfixed (1 downto -15) := to_sfixed(-1 ,1 ,-15 );
constant A2 : sfixed (0 downto 0) := to_sfixed(0 ,0 ,0 );

-- resultados parciales
signal sB0X0 : sfixed (15 downto -15);
signal sB1X1 : sfixed (15 downto -15);
signal sB2X2 : sfixed (15 downto -15);
signal sA1Y1 : sfixed (9 downto -15);
signal sA2Y2 : sfixed (9 downto -15);

-- salida en máxima resolucion
signal sY_max : sfixed (12 downto -15);
signal sY_sat : sfixed (8 downto -15);
```

Figura 4-5. Coeficientes del regulador del filtro RC sin Dither

En los casos con Dither se tiene la siguiente declaración de términos de la ecuación, representada en la siguiente figura.

```
architecture Behavioral of Regulador is

    --entrada y salida
    signal sX : sfixed (12 downto 0);
    signal sY : sfixed (12 downto 0);

    --valores anteriores de la entrada
    signal sX1 : sfixed (12 downto 0);
    signal sX2 : sfixed (12 downto 0);

    -- coeficientes
    constant B0 : sfixed (2 downto -15) := to_sfixed(0.0625*16*0.001 ,2 ,-15 );         --1/16
    constant B1 : sfixed (2 downto -15) := to_sfixed(0 ,2 ,-15 );       --(-0.99)/16
    constant B2 : sfixed (0 downto 0) := to_sfixed(0 ,0 ,0 );                  --0/16
    constant A1 : sfixed (1 downto -15) := to_sfixed(-1 ,1 ,-15);
    constant A2 : sfixed (0 downto 0) := to_sfixed(0 ,0 ,0 );

    -- resultados parciales
    signal sB0X0 : sfixed (15 downto -15);
    signal sB1X1 : sfixed (15 downto -15);
    signal sB2X2 : sfixed (13 downto 0);
    signal sA1Y1 : sfixed (14 downto -15);
    signal sA2Y2 : sfixed (13 downto 0);

    -- salida en máxima resolucion
    signal sY_max : sfixed (12 downto -15);
    signal sY_sat : sfixed (12 downto -15);
```

Figura 4-6. Coeficientes del regulador del filtro RC con Dither

### 4.2.2 El regulador del convertidor Buck

La ecuación en diferencias resultante en este caso es $\frac{0,083*(z-0,96)*(z-0,97)}{z*(z-1)}$, como se había comentado en el apartado anteriormente, aunque en este caso, el escalado de esta igualdad se ha hallado contrastando el resultado de los tiempos de establecimiento de la simulación en Matlab y el de Modelsim. Este factor se obtiene dividiendo el tiempo de Modelsim entre el de Matlab y estableciendo en VHDL el escalado de la ecuación y repitiendo el proceso hasta que los tiempos de establecimiento coinciden.

En VHDL, el regulador de los casos sin Dither para el convertidor Buck tendrán la siguiente declaración de términos de la ecuación, representada en la siguiente figura.

```
architecture Behavioral of Regulador is

    --entrada y salida
    signal sX : sfixed (12 downto 0);
    signal sY : sfixed (8 downto 0);

    --valores anteriores de la entrada
    signal sX1 : sfixed (12 downto 0);
    signal sX2 : sfixed (12 downto 0);

    -- coeficientes
    -- Las As y las Bs se meten tal cual
    constant B0 : sfixed (5 downto -15) := to_sfixed(0.083*12*1.4/16 ,5 ,-15 );
    constant B1 : sfixed (5 downto -18) := to_sfixed(-0.16019*12*1.4/16 ,5 ,-18 );
    constant B2 : sfixed (4 downto -30) := to_sfixed(0.0772896*12*1.4/16 ,4 ,-30 );
    constant A1 : sfixed (2 downto -15) := to_sfixed(-1 ,2 ,-15 );
    constant A2 : sfixed (2 downto 0) := to_sfixed(0 ,2 ,0 );

    -- resultados parciales
    signal sB0X0 : sfixed (18 downto -15);
    signal sB1X1 : sfixed (18 downto -18);
    signal sB2X2 : sfixed (17 downto -30);
    signal sA1Y1 : sfixed (15 downto -15);
    signal sA2Y2 : sfixed (15 downto 0);

    -- salida en máxima resolucion
    signal sY_max : sfixed (12 downto -15);
    signal sY_sat : sfixed (8 downto -15);
```

Figura 4-7. Coeficientes del regulador del convertidor Buck sin Dither

En los casos con Dither se obtiene la siguiente declaración de términos de la ecuación, representada en la siguiente figura.

```
architecture Behavioral of Regulador is

    --entrada y salida
    signal sX : sfixed (12 downto 0);
    signal sY : sfixed (12 downto 0);

    --valores anteriores de la entrada
    signal sX1 : sfixed (12 downto 0);
    signal sX2 : sfixed (12 downto 0);

    -- coeficientes
    -- Las As y las Bs se meten tal cual
    constant B0 : sfixed (5 downto -15) := to_sfixed(0.083*12*24.24/16 ,5 ,-15 );
    constant B1 : sfixed (5 downto -18) := to_sfixed(-0.16019*12*24.24/16 ,5 ,-18 );
    constant B2 : sfixed (4 downto -30) := to_sfixed(0.0772896*12*24.24/16 ,4 ,-30 );
    constant A1 : sfixed (2 downto -8) := to_sfixed(-1 ,2 ,-8 );
    constant A2 : sfixed (2 downto 0) := to_sfixed(0 ,2 ,0 );

    -- resultados parciales
    signal sB0X0 : sfixed (18 downto -15);
    signal sB1X1 : sfixed (18 downto -18);
    signal sB2X2 : sfixed (17 downto -30);
    signal sA1Y1 : sfixed (15 downto -23);
    signal sA2Y2 : sfixed (15 downto 0);

    -- salida en máxima resolucion
    signal sY_max : sfixed (12 downto -15);
    signal sY_sat : sfixed (12 downto -15);
```

Figura 4-8. Coeficientes del regulador del convertidor Buck con Dither

## 4.3 El módulo PWM

Como se ha visto anteriormente, la modulación PWM se basa en habilitar una señal, es decir, dar un '1' cuando la cuenta de un contador supera un valor determinado y deshabilitarlo en caso contrario. Esto genera una onda pseudo-analógica que permite controlar diversos dispositivos, como la tensión que aplica un filtro RC o un convertidor Buck, que son los casos que se están estudiando en este Trabajo.

El conjunto de entradas y salidas que tiene el módulo PWM es distinto para el filtro RC y para el Buck, ya que el convertidor necesita dos señales de modulación porque, al tener físicamente dos MOSFET de paso, se requiere una regulación de corrientes en dos sentidos distintos y de una serie de tiempos muertos que se consiguen mediante la desactivación de los dos transistores con el fin de anular dichas corrientes. Las señales que regulan este flujo se llaman HSM (High Side MOSFET) y LSM (Low Side MOSFET), siendo el que abre el camino a la corriente de entrada y el que la abre a tierra, respectivamente.

La siguiente figura muestra las entradas y salidas del módulo PWM diseñado para el filtro RC. En este caso, es la cabecera para un caso con Dither porque el ciclo de trabajo tiene 12 bits (8 de consigna y 4 de acumulación), aunque en un caso sin Dither, sería tan sencillo como poner que el ciclo de trabajo tiene sólo 8 bits, eliminando así los del acumulador.

```
entity GestorPWM is
    Port ( CicloTrabajo : in  STD_LOGIC_VECTOR (11 downto 0);
           Clk : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           CE : in std_logic;
           Pwm : out  STD_LOGIC);
end GestorPWM;
```

Figura 4.9. Entidad genérica del módulo PWM con Dither RC

Seguidamente, las entradas y salidas del módulo PWM diseñado para el convertidor Buck, común en los dos casos. Como hemos dicho anteriormente, este caso se corresponde con un caso con Dither porque el ciclo de trabajo tiene 12 bits, pero la cabecera de uno sin Dither tendría el ciclo de trabajo con 8 bits.

```
entity GestorPWM is
    Port ( CicloTrabajo : in  STD_LOGIC_VECTOR (11 downto 0);
           Clk : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           CE : in std_logic;
           LSM : out  STD_LOGIC;
           HSM : out  STD_LOGIC);
end GestorPWM;
```

Figura 4.10. Entidad genérica del módulo PWM con Dither Buck

También se conoce el problema que supone las oscilaciones producidas por el ciclo límite, por lo que a continuación se va a explicar cómo implementar la solución propuesta, el Dither, tanto en una planta compuesta por un filtro RC y otra que es un convertidor Buck.

### 4.3.1 PWM sin la mejora del Dither

En el caso del filtro RC, se trata de un módulo PWM tal y como lo es conocido: una única señal para generar la modulación correspondiente. La siguiente figura mostrará su implementación en VHDL.

```
begin

    if (Reset = '1') then
        contador := (others => '0');
        Pwm <= '0';
    else

        if (rising_edge (Clk)) then
            contador := contador + 1;

            if (contador < unsigned(CicloTrabajo)) then
                Pwm <= '1';
            else
                Pwm <= '0';
            end if;
        end if;
    end if;
end process;
```

Figura 4-11. Implementación del PWM sin Dither del filtro RC

En cambio, a la hora de diseñar el módulo PWM para el convertidor Buck, se tuvo que tener en cuenta que este necesita dos señales que generen las correspondientes modulaciones de forma sincronizada, tal y como se había visto. Por un lado, debe haber un contador de referencia para activar las señales de control. Por otro lado, tenemos la generación en sí de las señales, en las que se tiene que tener en cuenta los tiempos muertos, es decir, cuando ambos MOSFET están en corte, y que las señales HSM y LSM no se habilitan a la vez, lo que produciría un cortocircuito. La siguiente figura muestra el proceso:

```
conTmuertoUp <= std_logic_vector(unsigned(tiempoMuerto) + unsigned(CicloTrabajo));

dead_time_prepare: process(Clk, Reset)
begin
    if (Reset = '1') then
        HSM <= '0';                                          -- el inicio de la actuac
        LSM <= '0';                                          -- el inicio de la actuac

    elsif (rising_edge (Clk)) then
    --control de HSM
        if(contador < unsigned(CicloTrabajo)) then
            HSM <= '1';
        else
            HSM <= '0';
        end if;

    --control de LSM
        if((contador < unsigned(conTmuertoUp)) or (CicloTrabajo > conTmuertoDown)) then
            LSM <= '0';
        elsif(contador < unsigned(conTmuertoDown)) then
            LSM <= '1';
        else
            LSM <= '0';
        end if;

    end if; -- if inicial
end process;            --dead_time_prepare
```

Figura 4-12. Implementación del PWM sin Dither del convertidor Buck

Cabe destacar que la señal “conTmuertoUp” limita la señal HSM para generar el primer tiempo muerto y la señal “conTmuertoDown”, análoga a la primera, limita a LSM para dar de nuevo un descanso al convertidor.

### 4.3.2 PWM con la mejora del Dither

Tanto para el filtro RC como para el convertidor Buck, se trata de un PWM variable entre dos valores sucesivos, es decir, en algunos ciclos de conmutación, la consigna será N y en otros será N+1. Este cambio depende de los bits de acumulación, que serán los 4 LSB de la

señal “CicloTrabajo” (como ya se había comentado). El proceso de asignación de la consigna viene dado por nuevamente por un contador y por la señal “CicloTrabajo”, que en vez de tener 8 bits como en la versión normal, tendrá 12, de los que se distinguen entre bits de consigna (los 8 MSB) y los de acumulación (los 4 LSB). Los de acumulación están directamente conectados con un acumulador de 5 bits cuya misión será ir sumando el valor de estos bits al valor que tenía en el anterior ciclo de conmutación. Es destacable que tiene 5 bits para evitar la sobrecarga y, además, el quinto bit sirve a modo de señalizador para cuando este supere el valor máximo, que es 16 (“10000” en binario). Cuando el acumulador supera este valor, es decir, el MSB es ‘1’, este se limpia (pasa a ser ‘0’) y se suma “1” al ciclo de trabajo, haciendo así que la consigna del PWM varíe entre dos valores consecutivos.

La diferencia entre los módulos PWM del filtro y del convertidor es que el filtro utiliza una señal y el convertidor dos y, para ese último, el Dither se aplicará a HSM por ser la que controla la tensión a la salida del sistema.

La siguiente figura muestra la implementación del Dither para el filtro RC:

```
elsif(rising_edge(Clk)) then
    contador := contador + 1;

    if(CE = '1') then
        cicle <= CicloTrabajo(11 downto 4);
        resto <= CicloTrabajo(3 downto 0);
        sumador <= std_logic_vector(unsigned(sumador) + unsigned(CicloTrabajo(3 downto 0)));

    elsif((sumador(4) = '1') and (unsigned(cicle) < 255)) then
        cicle <= std_logic_vector(unsigned(cicle) + unsigned(uno));
        sumador(4) <= '0';
    end if;

    if (contador < unsigned(cicle)) then
        Pwm <= '1';
    else
        Pwm <= '0';
    end if;
end if;
```

Figura 4-13. Implementación del PWM con Dither del filtro RC

A continuación, se verá en la imagen el código VHDL utilizado en el proceso del Dither del convertidor Buck:

```
elsif(rising_edge(Clk)) then
    contador := contador + 1;

    if(CE = '1') then
        cicle <= CicloTrabajo(11 downto 4);
        resto <= CicloTrabajo(3 downto 0);
        sumador <= std_logic_vector(unsigned(sumador) + unsigned(CicloTrabajo(3 downto 0)));

    elsif((sumador(4) = '1') and (unsigned(cicle) < 255)) then
        cicle <= std_logic_vector(unsigned(cicle) + unsigned(uno));
        sumador(4) <= '0';
    end if;

    --control de HSM
    if(contador < unsigned(cicle)) then
        HSM <= '1';
    else
        HSM <= '0';
    end if;

    --control de LSM
    if((contador < unsigned(conTmuertoUp)) or (cicle > conTmuertoDown)) then
        LSM <= '0';
    elsif(contador < unsigned(conTmuertoDown)) then
        LSM <= '1';
    else
        LSM <= '0';
    end if;

end if;
```

Figura 4-14. Implementación del PWM con Dither del convertidor Buck

## 4.4 Implementación del sistema: el controlador

Como ya se ha visto, el controlador implementa la totalidad del sistema: el gestor del ADC, el regulador, el módulo PWM y el cálculo del error. En VHDL, se trata de una entidad que engloba al resto mediante el uso de las instrucciones “component” y “port map”, que permiten unir los bloques en el controlador, aunque el controlador no es solo la unión de estos, sino que también incluye al cálculo de error, la captura de la consigna y el modo de operación del sistema, siendo los dos últimos utilizados únicamente a la hora de depurar el funcionamiento del controlador.

El cálculo del error es el componente que resta la lectura del ADC con la consigna a la que se quiere llegar. Esta línea es para un caso con 7 bits de ADC, pero para hacer el resto, es tan sencillo como quitar los ceros de relleno e incluir los LSB del ADC. La siguiente figura muestra el código de su implementación:

```
-- Cálculo del error
error <= std_logic_vector(signed('0' & consignaRegistrada) - signed('0' & adcRegistrado(11 downto 5) & "00000"));
```

Figura 4-15. Cálculo del error

Es destacable que esta línea será diferente para cada tipo de ADC utilizado. En este Trabajo, se han utilizado para el estudio del ciclo límite controladores con un ADC de 7 bits, 10 y 12, con y sin Dither, siendo cada controlador independiente del resto, es decir, se tendrá un controlador de 7 bits de ADC con Dither, otro de 7 bits sin Dither, de 12 bits de ADC sin Dither, y así sucesivamente.

En cuanto al modo de operación, el sistema implementado cuenta con un interruptor para activar la opción “Swap”, la cual permite la depuración del controlador en lazo abierto, es decir, posibilita la depuración del regulador y el módulo PWM sin la necesidad de conectar

el gestor del ADC a la salida del sistema, obligando así que la consigna del PWM sea la que se introduce con los interruptores. La siguiente imagen muestra el código que describe este modo de funcionamiento:

```
-- Multiplexor de entrada al GestorPWM
cicloTrabajo <= actuacion(7 downto 0) when Swap='0' else Consigna;
```

Figura 4-16. Selector del modo de operación

## 4.5 Filtro RC

Para poder probar el filtro RC en una simulación, primero se tiene que hacer un modelo en VHDL de la planta que emule su comportamiento. Las entradas que tiene es la señal PWM y una señal que indica si la simulación sigue siendo efectiva; y la salida es la tensión de salida del sistema. La siguiente figura muestra la lógica que sigue el filtro RC en su versión hardware:

```
--Proceso para simular la intensidad y la tensión.
-- La intensidad será igual a V/R, donde V es la tensión del ciclo de simulación anterior.
simulacion_corriente : process
  variable Iv : real :=0.0; -- Se podría hacer que la simulación arrancara en otro punto
  variable Vv : real :=0.0; -- Ídem
begin
  if(PWM='1') then
    Iv:=(Von-Vv)/R;
  else
    Iv:=-Vv/R;
  end if;
  Vv:=Vv+Iv*(At/C);

  I<=Iv;
  VOut<=Vv;
  if end_sim then
      wait;
  else
      wait for At_t;
  end if;

end process;
```

Figura 4-17. Modelo VHDL del filtro RC

La simulación no solo se basa en VHDL, sino que también es necesaria una herramienta de Matlab llamada “sisotool”, tal y como se había comentado anteriormente, la cual nos proporciona de mediante un pequeño código y la ecuación en diferencias del filtro su respuesta al escalón y, con ello, su tiempo de establecimiento (de unos 3,5ms), mostrado en la siguiente imagen:

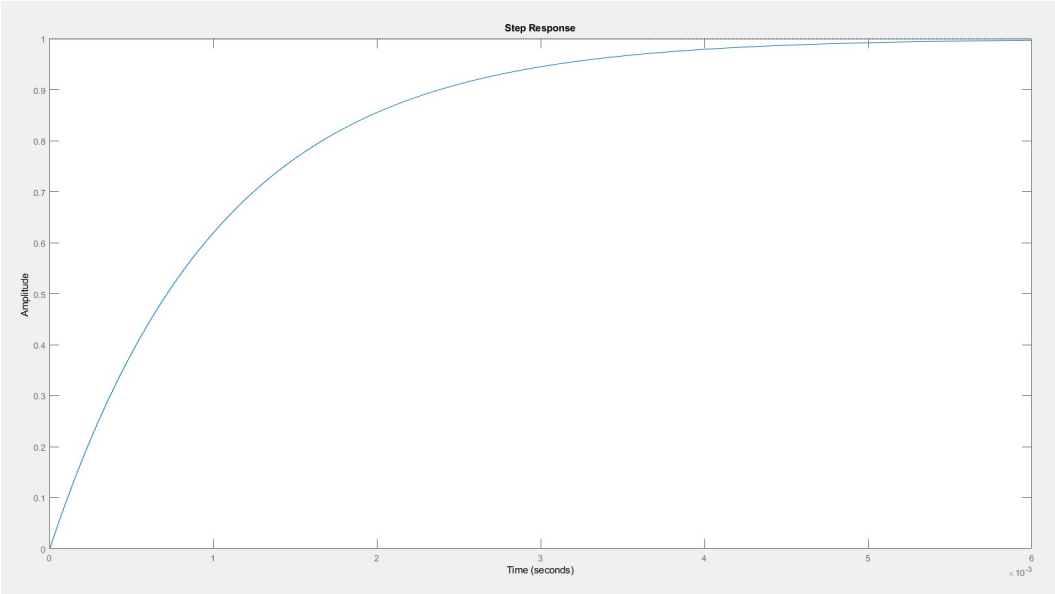


Figura 4-18. Tiempo de establecimiento del filtro RC

## 4.6 Convertidor Buck

Al igual que se ha hecho con el filtro RC, se ha diseñado un modelo en VHDL específico para el convertidor Buck, aunque su cabecera es más compleja en cuanto a señales. La siguiente figura las mostrará:

```
entity BuckMonofaseReal is
    port (
        --In
        Clk : in std_logic;
        Reset : in std_logic;
        HSM : in std_logic; -- On = '1', off = '0'
        LSM : in std_logic; -- On = '1', off = '0'
        Vg : in real;
        Ir : in real;
        -- Out
        IL : out real;
        Vout : out real
    );
end BuckMonofaseReal;
```

Figura 4-19. Entidad del modelo del convertidor Buck

Es destacable un reset asíncrono, es decir, independiente de los flancos de reloj, las señales de control mediante PWM HSM y LSM, de las que ya se ha hablado, “Vg” como la tensión de entrada del sistema (fijada a 12V), “Ir” e “IL” como las corrientes de entrada y salida, respectivamente, y “Vout” como la tensión de salida.

La siguiente imagen muestra el código que implementa la emulación del convertidor:

```
IL <= iLAux;
Vout <= voutAux;

SWITCHMUX: process(HSM, LSM, Vg, Ir, iLAux, voutAux)
-- Selection (multiplexer) of values to be added to input current and output voltage
begin
voutAuxAdd <= (iLAux - Ir);
    if HSM = '1' and LSM = '0' then -- 1,0
        iLAdd <= (Vg - voutAux);
    elsif HSM = '0' and LSM = '1' then -- 0,1
        iLAdd <= -voutAux;
elsif HSM = '0' and LSM = '0' then -- 0,0
  if iLAux > 0.0 then
    iLAdd <= -voutAux;
  else
    iLAdd <= (Vg - voutAux);
  end if;
else -- 1,1
  report "Cagada, no pueden estar los dos interruptores cerrados";
    end if;
end process SWITCHMUX;


DIFFEQ: process(Clk, Reset)
-- Update of Vout and Iin each clock cycle
begin
    if Reset = '1' then
        voutAux <= VOINIT;
        iLAux <= ILINIT;
    elsif rising_edge(Clk) then
        iLAux <= iLAux + iLAdd*dtL;
        voutAux <= voutAux + voutAuxAdd*dtC;
    end if;
end process DIFFEQ;
```

Figura 4-20. Modelo VHDL del convertidor Buck

Nuevamente, también se ha utilizado la aplicación de Matlab para obtener el tiempo de establecimiento de la planta:

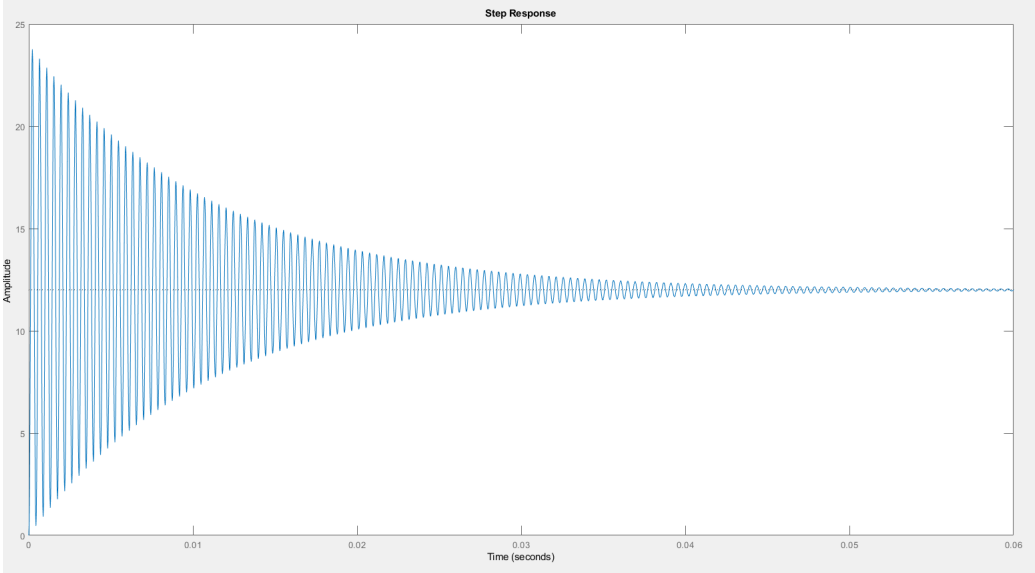


Figura 4-21: Tiempo de establecimiento del convertidor Buck

Para terminar, es destacable la implementación de un divisor de tensión en el lazo de alimentación del ADC porque el valor máximo de tensión de salida del Buck es de 12V y la tensión máxima de entrada del chip es de 5V. Se trata de un divisor formado por dos resistencias, tal y como muestra la siguiente figura:

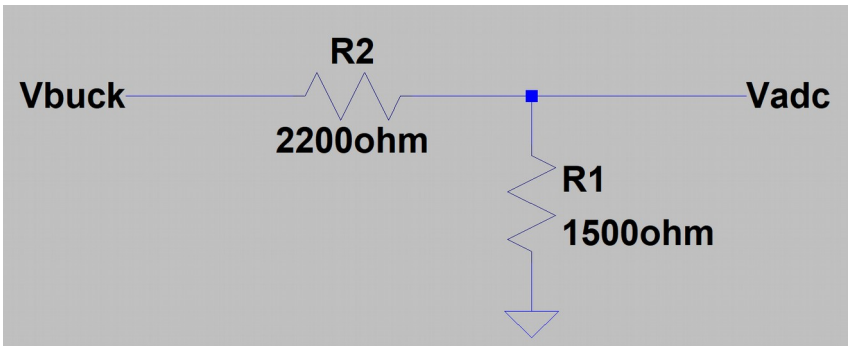


Figura 4-22: Esquema del divisor de tensión del Buck

# 5 Integración, pruebas y resultados

Tras haber descrito mediante VHDL todos los bloques que componen el sistema, ahora es momento de diseñar un fichero de pruebas en este lenguaje llamado testbench con el objetivo de probar cada controlador antes de su integración en la placa contenedora de la FPGA. Después de simular, se cargará el programa en hardware real y se contrastarán los resultados simulados en Modelsim con los obtenidos experimentalmente.

## *5.1 Diseño de los ficheros de simulación*

Un fichero de simulación o testbench es una entidad sin puertos, hablando en VHDL, la cual instancia al componente que se quiere probar mediante estímulos o instrucciones que emulen un comportamiento real.

En todos los testbench utilizados se instancia al controlador y al modelo de la planta gracias a las instrucciones "component" y "port map" y se implementa un proceso que actuará como un ADC de 12 bits con las mismas características que el real. Estos ficheros son idénticos para todos los casos de ADC, sea con o sin Dither, y para cualquier planta con la salvedad del escalado en la señal "adc", que simula la tensión que lee el ADC, ya que esta depende de la tensión leída y el valor máximo que puede leer el dispositivo, que en este trabajo es 5V.

La siguiente figura muestra el código de este proceso para el filtro RC:

```
-- Proceso de simulación del ADC
 adc_proc : process
 --variable tmpVanalog : std_logic_vector (11 downto 0) := (others => '0');
 variable capturaTension : real := 0.0;
 begin
     adc <= (others => 'Z');
     while not end_sim loop
         wait until nCs = '0' and nRD = '0';
         wait for 80 ns;                                  --t2 (rd to busy)
         nBUSY <= '0';
         capturaTension := vanalogica;
         wait for 3 us;                                   --tconv;
         nBUSY <= '1';
         wait for 100 ns;                                 --t6 (tiempo de estabilizacion del dato)

         -- el dato esta listo
         if (capturaTension > 3.3) then
             adc <= x"FFF";
         elsif capturaTension < 0.0 then
             adc <= x"000";
         else
             adc <= STD_LOGIC_VECTOR(CONV_UNSIGNED(INTEGER(819.0*capturaTension),12));   --4095/5=819
         end if;

         wait until nRD = '1';                            --liberar bus de datos
         wait for 100 ns;                                 --relinquish time (liberacion del bus)
         adc <= (others => 'Z');
     end loop;
 wait;
 end process;
```

Figura 5-1: Proceso de emulación del ADC del filtro RC

A continuación, para el convertidor Buck:

```
Proceso de simulación del ADC
 adc_proc : process
 --variable tmpVanalog : std_logic_vector (11 downto 0) := (others => '0');
 variable capturaTension : real := 0.0;
 begin
     adc <= (others => 'Z');
     while not end_sim loop
         wait until nCs = '0' and nRD = '0';
         wait for 80 ns;                                  --t2 (rd to busy)
         nBUSY <= '0';
         capturaTension := vanalogica;
         wait for 3 us;                                   --tconv;
         nBUSY <= '1';
         wait for 100 ns;                                 --t6 (tiempo de estabilizacion del dato)

         -- el dato esta listo
         if (capturaTension > 3.3) then
             adc <= x"FFF";
         elsif capturaTension < 0.0 then
             adc <= x"000";
         else
             adc <= STD_LOGIC_VECTOR(CONV_UNSIGNED(INTEGER(341.0*capturaTension),12));   --4095/12=341
         end if;

         wait until nRD = '1';                            --liberar bus de datos
         wait for 100 ns;                                 --relinquish time (liberacion del bus)
         adc <= (others => 'Z');
     end loop;
 wait;
 end process;
```

Figura 5-2: Proceso de emulación del ADC del convertidor Buck

Para finalizar, estos ficheros incluyen otro proceso (sin contar con la generación de reloj) que simula el comportamiento que tendría el usuario como probador del prototipo. Es el proceso de selección de consigna, el cual impone una consigna para ver que el sistema devuelve el valor correcto en lazo cerrado.

## *5.2 Resultados simulados y experimentales*

Tras finalizar y crear un proyecto de simulación en Modelsim, es hora de simular. Hay que tener en cuenta que el tiempo de actuación de las consignas, ya que el error del sistema se mantiene estable, pero no nulo, de forma temporal debido a que la actuación varía en unos pocos decimales cada ciclo de conmutación. Cuando esta alcanza un valor que hace que el error varíe, es cuando se hace visible la oscilación del ciclo límite. La siguiente figura muestra el fenómeno descrito:

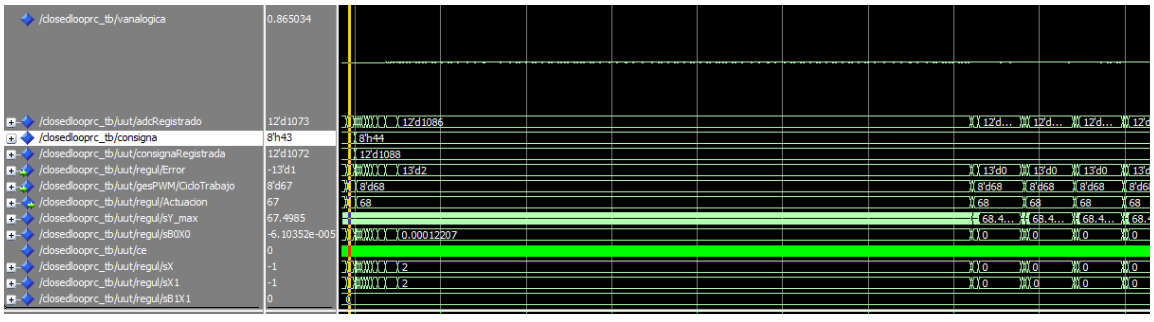


Figura 5-3: Transición del error estable temporalmente a ciclo límite

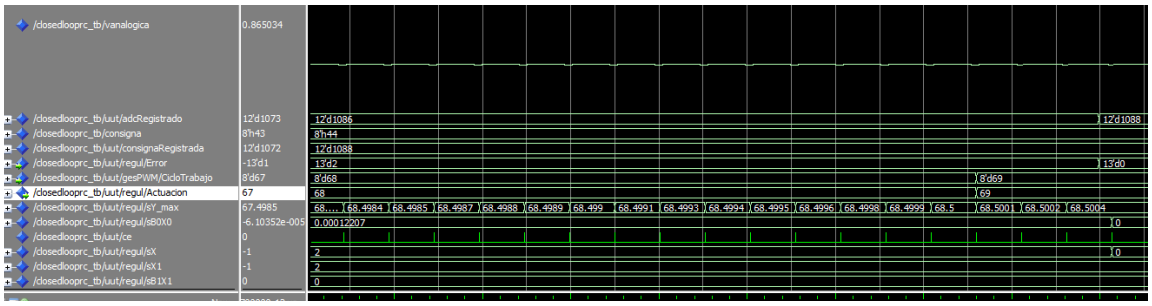


Figura 5-4: Zoom de la actuación en la transición

En estas imágenes, se ve la transición para el filtro RC en el caso de ADC de 12 bits sin Dither y este tiempo es de 35,963 ms.

A la hora de contrastar, ya que el ciclo límite es un fenómeno aleatorio, se han hecho medidas con diferentes consignas, por lo que aparecerán por cada caso de ADC dos consignas distintas.

### 5.2.1 Resultados del filtro RC

Para el caso del ADC con 7 bits útiles, no se verá LCO en ningún caso, haya Dither o no, debido a que tenemos 1 bit más de PWM que de ADC, lo que hace imposible que no haya una actuación para una lectura en concreto. Esta teoría se corrobora con las siguientes figuras:

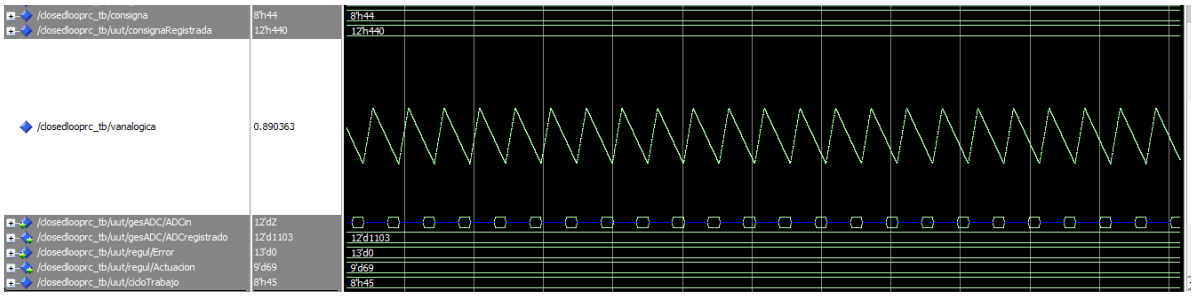


Figura 5-5: Caso de 7 bits de ADC sin Dither en simulación RC

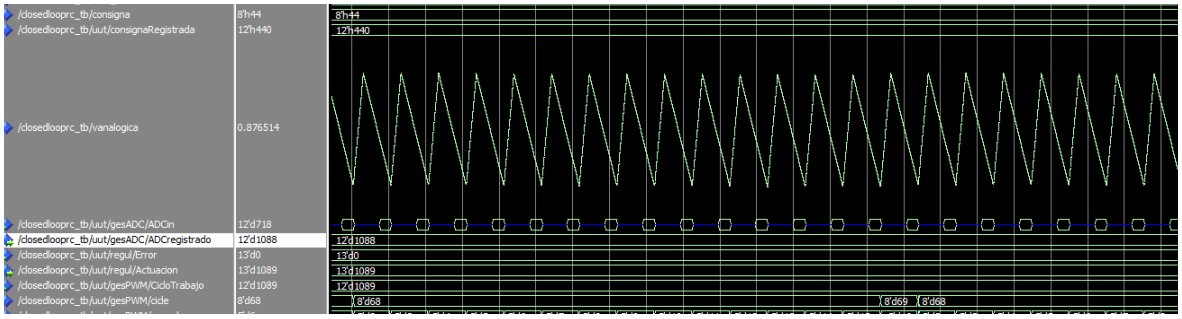


Figura 5-6: Caso de 7 bits de ADC con Dither en simulación RC

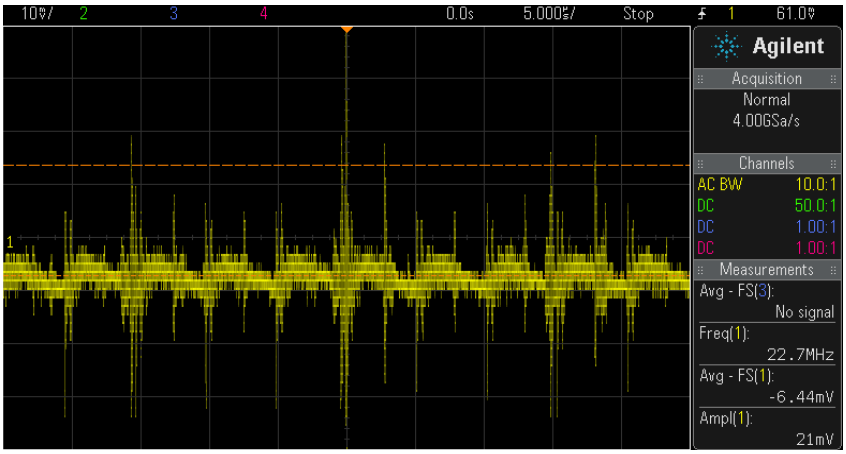


Figura 5-7: Caso de 7 bits de ADC sin Dither en la práctica RC

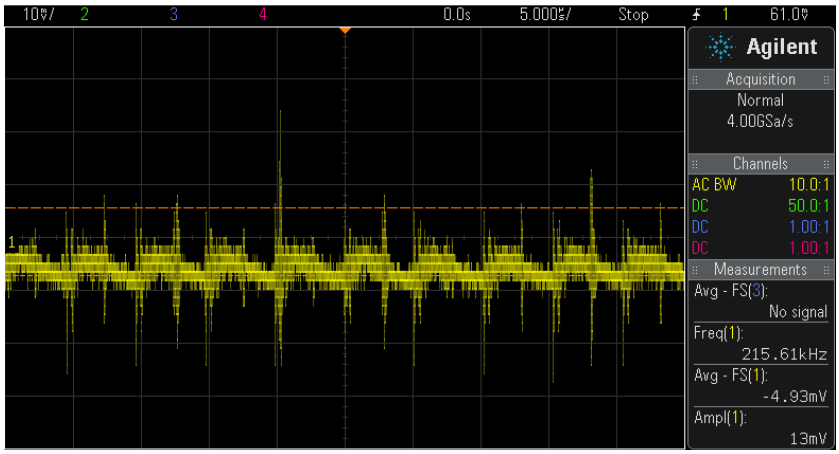


Figura 5-8: Caso de 7 bits de ADC con Dither en la práctica RC

Podemos confirmar que no hay ciclo límite porque la actuación se mantiene estable, por lo que solo vemos rizado de conmutación. Como era de esperar, el error es nulo gracias al integrador del regulador.

Para los experimentos con 10 bits de ADC sin Dither, se tendrán 2 bits más de ADC que de PWM, por lo que en esta situación sí habrá oscilaciones indeseadas, aunque no tienen por qué existir para todas las consignas posibles. En cambio, cuando se aplica el Dither convertimos los 4 LSB de la actuación del regulador a PWM, es decir, es como si se tuviera un módulo PWM de 8 + 4 bits de resolución, siendo un valor superior al número de bits del ADC, por lo que no habrá ciclo límite. Las siguientes imágenes lo demuestran:

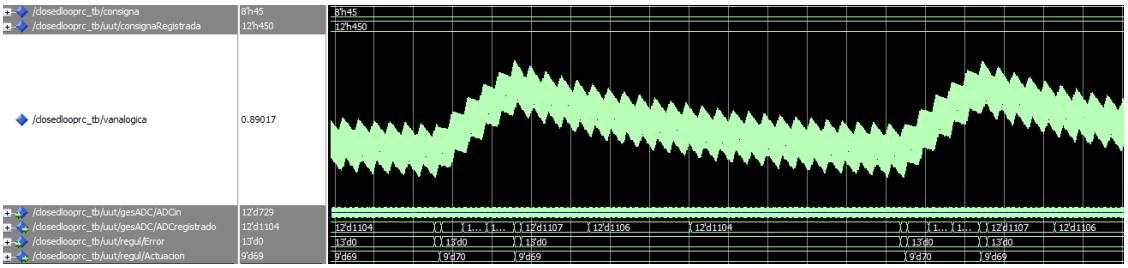

Figura 5-9: Caso de 10 bits de ADC sin Dither en simulación (periodo) RC

Podemos apreciar que debido a un error de pintado de Modelsim, no se puede observar bien la forma que tendría la oscilación del ciclo límite, pero haciendo zoom sobre un periodo vemos lo siguiente:

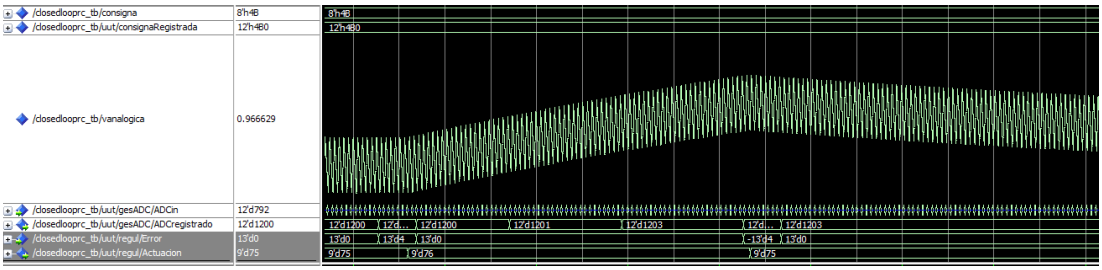


Figura 5-10: Caso de 10 bits de ADC sin Dither en simulación (amplitud) RC

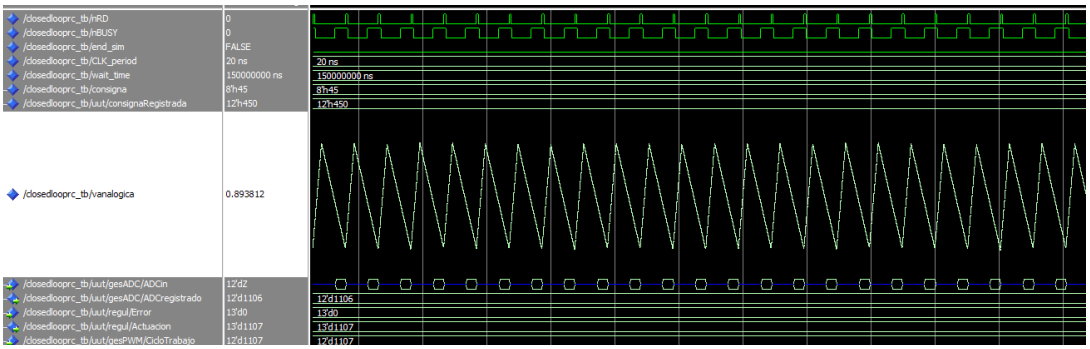


Figura 5-11: Caso de 10 bits de ADC con Dither en simulación RC

Debido a que la resolución del osciloscopio no es lo suficientemente pequeña como para apreciar la oscilación vista en simulación, solo se ve el rizado de conmutación. Más adelante, se verá que los valores del rizado del LCO son mucho menores que la resolución del osciloscopio.

Para los casos de 12 bits de ADC sin Dither, es altamente probable encontrar una consigna que genere ciclo límite a la salida debido a que el ADC tiene 4 bits de resolución más que el PWM. En cambio, si se introduce la técnica del Dither, tendremos 12 bits virtuales de PWM y 12 de ADC, por lo que puede que el LCO sea eliminado totalmente o verse atenuado y a una frecuencia mayor que la de la oscilación. Las siguientes figuras muestran los resultados medidos de este caso:

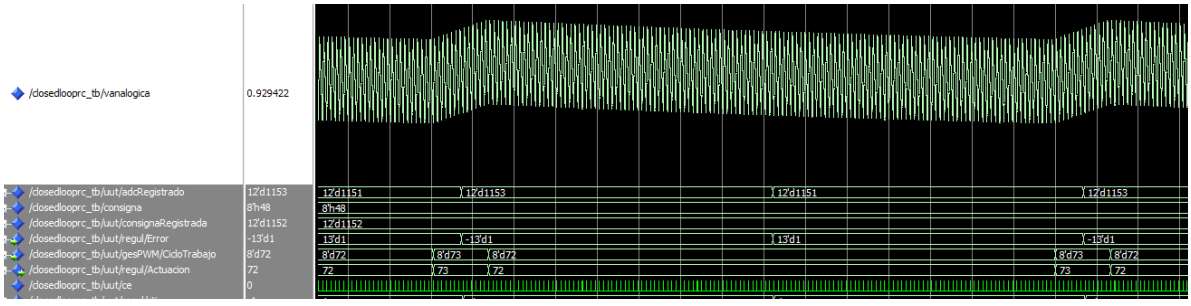

Figura 5-12: Caso de 12 bits de ADC sin Dither y consigna 0x48 en simulación RC

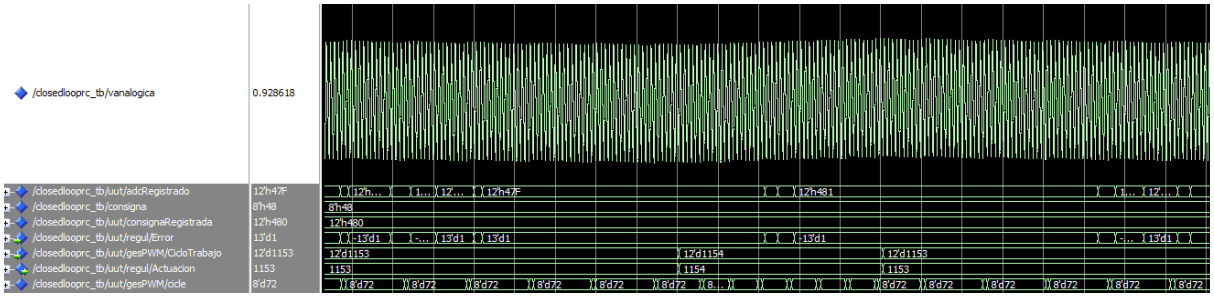


Figura 5-13: Caso de 12 bits de ADC con Dither y consigna 0x48 en simulación RC

Para esta consigna se puede observar que, pese a utilizar la técnica del Dither, sigue habiendo oscilación indeseada, aunque se trata de rizado de Dither y su amplitud es mucho menor a la del ciclo límite, pero su frecuencia es muy superior (se verá en el apartado 5.3).

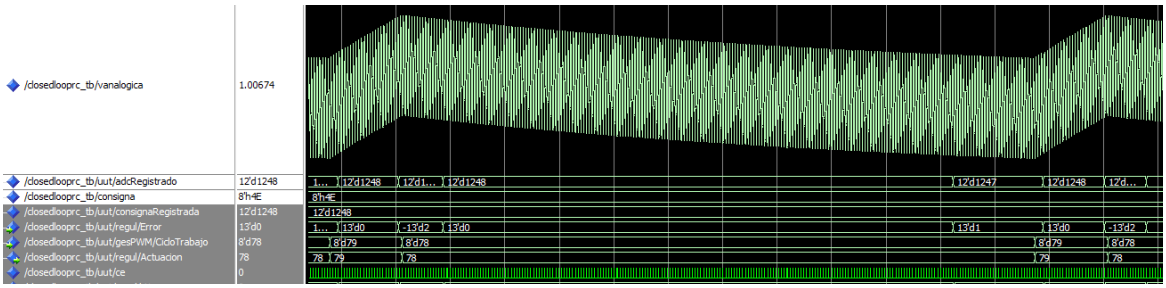


Figura 5-14: Caso de 12 bits de ADC sin Dither y consigna 0x4E en simulación RC

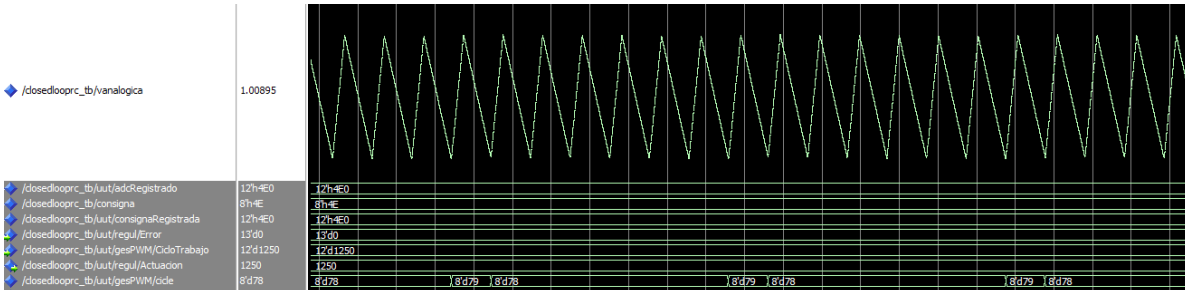


Figura 5-15: Caso de 12 bits de ADC con Dither y consigna 0x4E en simulación RC

En cambio, en esta consigna el efecto de la oscilación indeseada se elimina completamente, dejando la actuación estable. Este hecho sostiene que en algunos casos el ciclo límite se elimina empleando Dither en el PWM.

### 5.2.2 Resultados del convertidor Buck

Al igual que se ha hecho con el filtro RC, se mostrarán imágenes de las simulaciones y pruebas experimentales de las mismas situaciones de ADC.

Para el caso de 7 bits de ADC se tendrán los siguientes resultados:

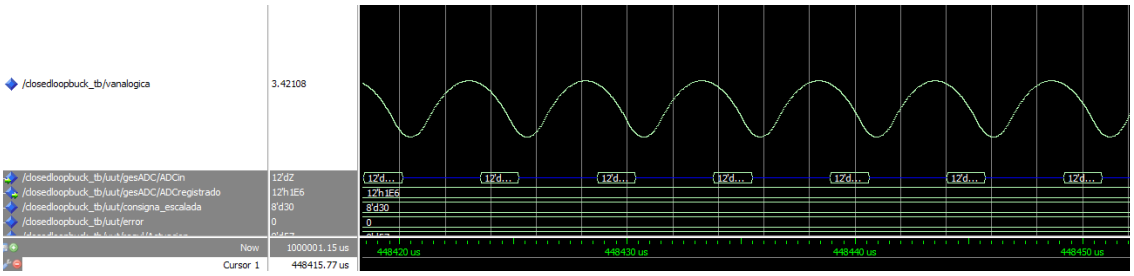


Figura 5-16: Caso de 7 bits de ADC sin Dither en simulación Buck

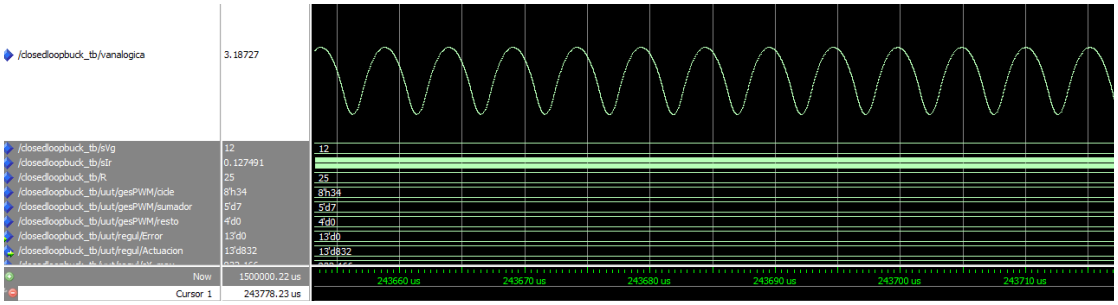


Figura 5-17: Caso de 7 bits de ADC con Dither en simulación Buck

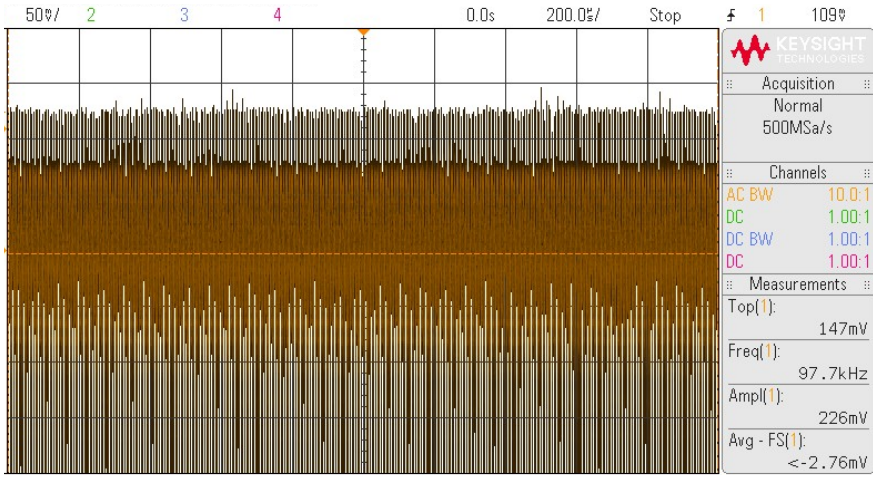


Figura 5-18: Caso de 7 bits de ADC sin Dither en la práctica Buck

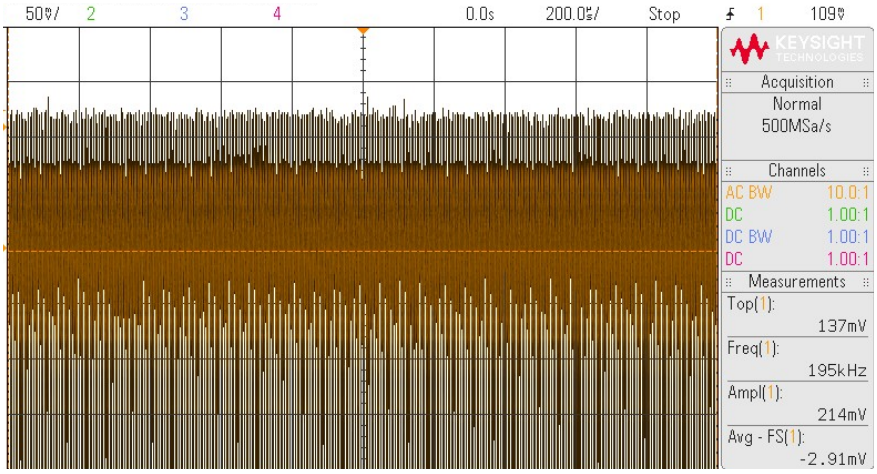


Figura 5-19: Caso de 7 bits de ADC con Dither en la práctica Buck

De nuevo, se ve que cuando la resolución del ADC es menor que la del PWM no hay ciclo límite, por lo que solo se puede apreciar el rizado de conmutación, que para el convertidor Buck será de 1,7mV (en simulación) aproximadamente.

Para el caso de 10 bits de ADC, aparecerá ciclo límite cuando no haya Dither y se eliminará cuando se le aplique la técnica, aunque, por el contrario, aparecerá rizado de Dither. Las siguientes figuras lo corroboran:

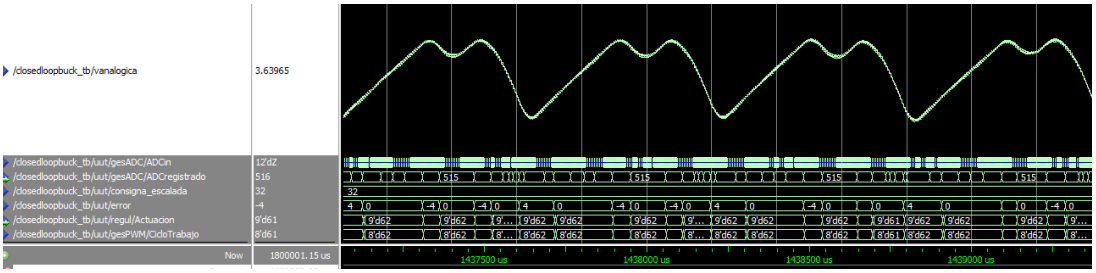


Figura 5-20: Caso de 10 bits de ADC sin Dither en simulación Buck

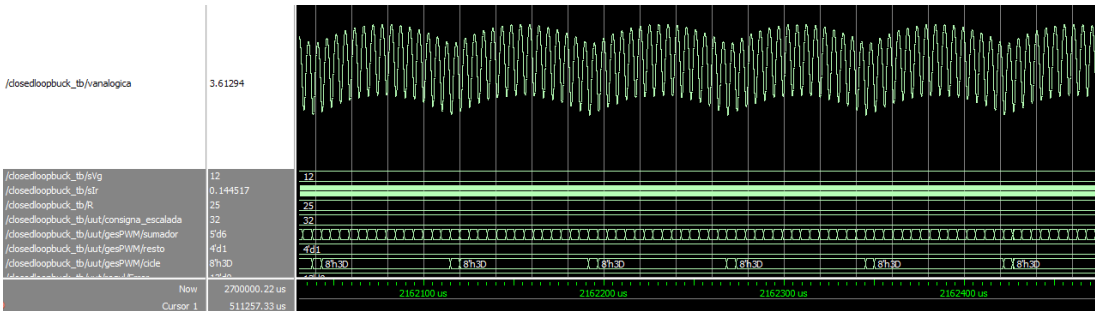


Figura 5-21: Caso de 10 bits de ADC con Dither en simulación Buck

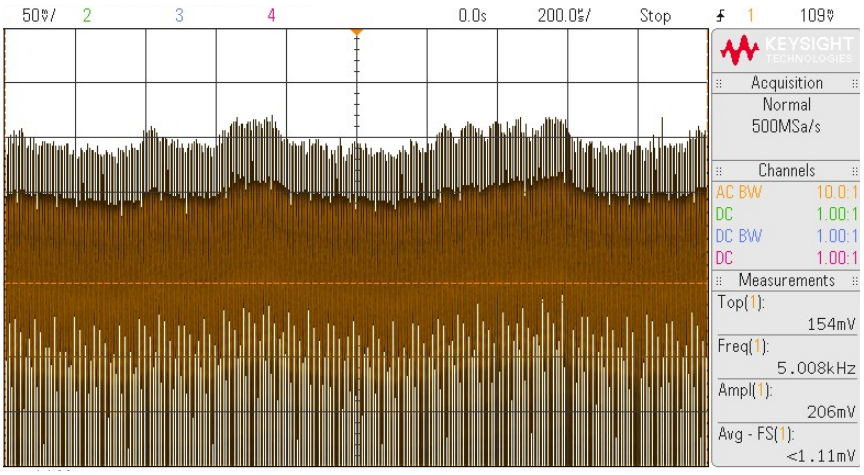


Figura 5-22: Caso de 10 bits de ADC sin Dither en la práctica Buck

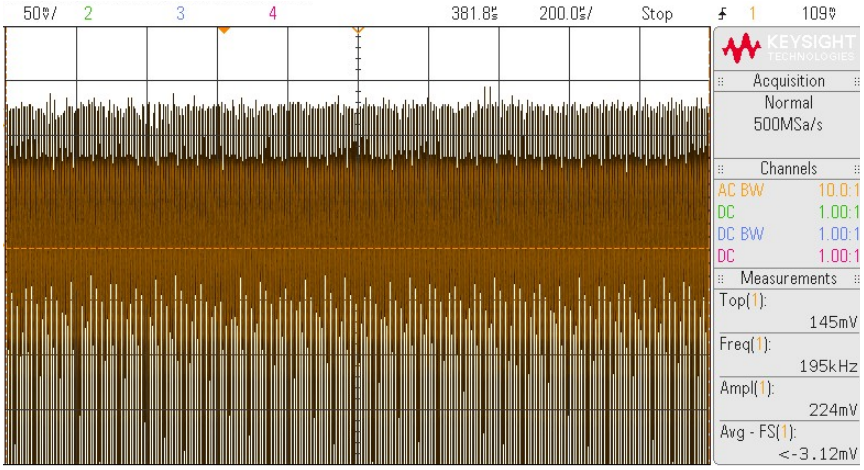


Figura 5-23: Caso de 10 bits de ADC con Dither en la práctica Buck

Cabe destacar que el ciclo límite se atenúa mucho en simulación y poco en la práctica respecto al rizado de conmutación. Esto se debe a que en simulación los condensadores son ideales, es decir, no tienen resistencia parásita. En cambio, en la realidad y al haber utilizado condensadores electrolíticos, esta resistencia parásita no es despreciable y eso hace que el rizado de conmutación real sea mucho mayor que el de simulación (180mV frente a 1,7mV, aproximadamente). Por tanto, el rizado de Dither de la prueba experimental, mostrado en la figura 5-23, es tan pequeño que no se puede ver gracias a que la resolución del osciloscopio es de 10mV por división y este rizado tendrá mucha menor amplitud.

Para el caso de 12 bits de ADC, aparecerá ciclo límite cuando no haya Dither y se atenuará cuando se le aplique la técnica, aunque, por el contrario, volverá a aparecer rizado de Dither. Las siguientes figuras lo demuestran:

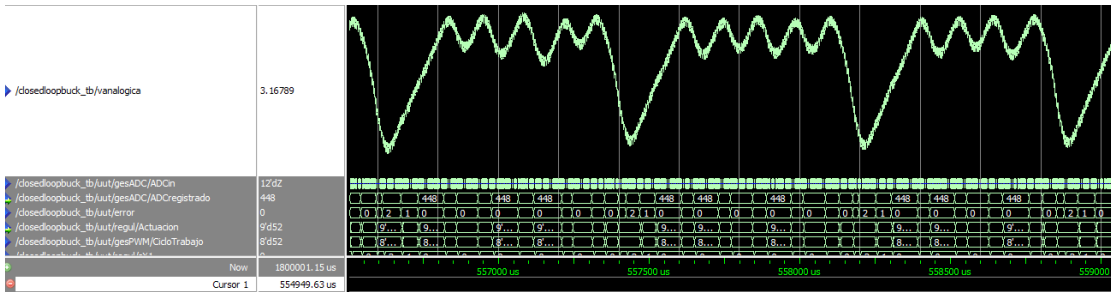


Figura 5-24: Caso de 12 bits de ADC sin Dither en simulación Buck

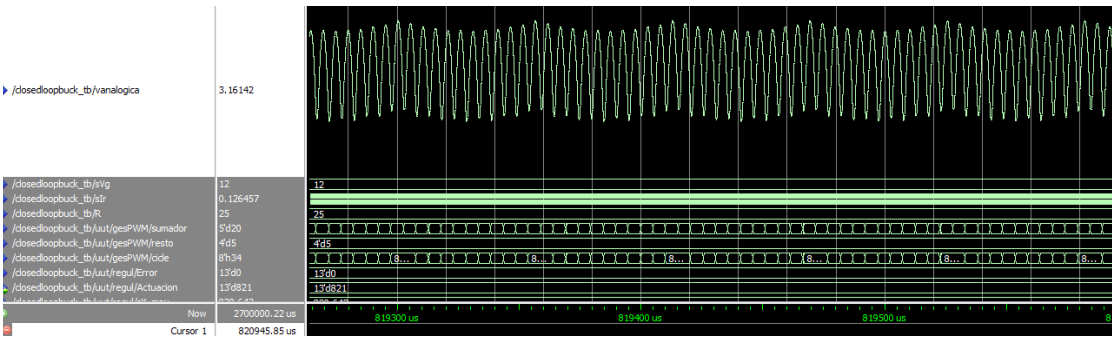


Figura 5-25: Caso de 12 bits de ADC con Dither en simulación Buck

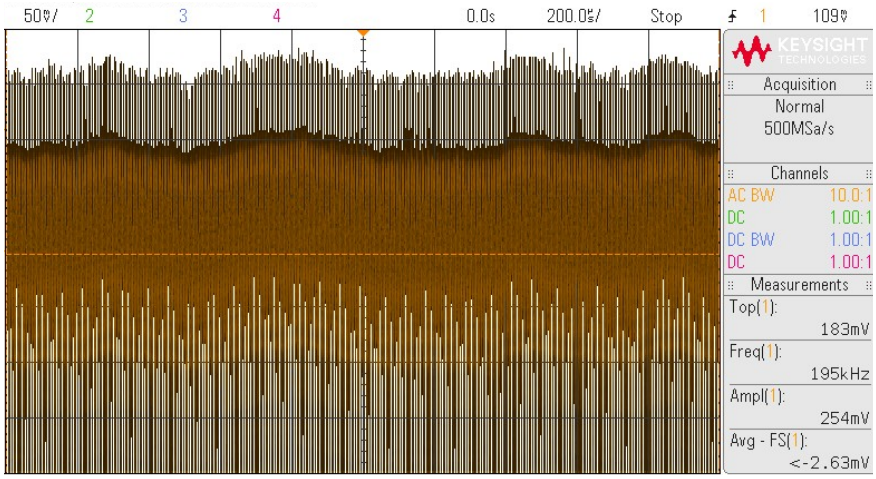


Figura 5-26: Caso de 12 bits de ADC sin Dither en la práctica Buck

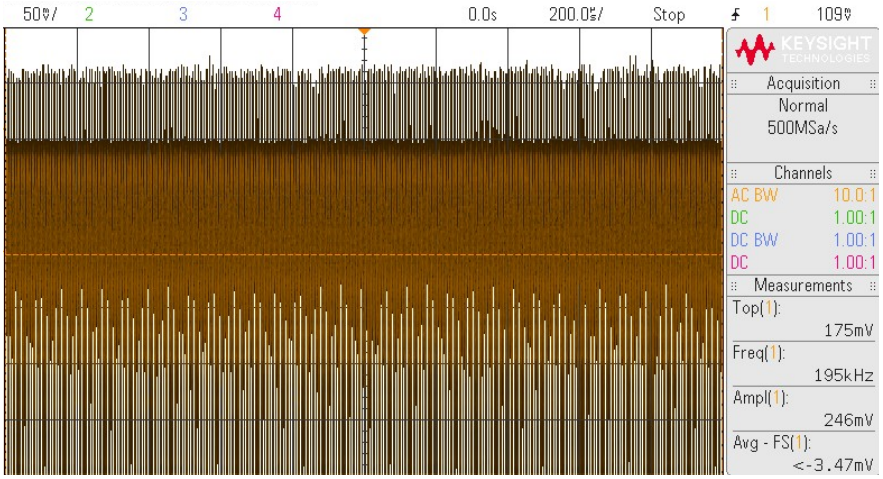


Figura 5-27: Caso de 12 bits de ADC con Dither en la práctica Buck

En este caso, se puede apreciar cómo el rizado acoplado de LCO a la señal de salida cuando no se ha aplicado Dither es relativamente grande. Nuevamente, al implementar la técnica se ve cómo ese rizado se atenúa.

## *5.3 Comparación de los casos con Dither y sin Dither*

Tras observar las capturas anteriores referidas a los casos de 7, 10 y 12 bits de resolución de ADC, es fácil hacerse a la idea del valor de los rizados teniendo en cuenta la escala de tensión en el caso de las pruebas experimentales, pero en las simulaciones no hay escala, por lo que se ha medido y apuntado los valores de amplitud, frecuencia y caso de ADC. La siguiente tabla muestra los datos de simulación del filtro RC:

| **Caso (consigna)** | **Amp. LCO (mV)** | **Frec. LCO (Hz)** | **Amp. Dither (mV)** | **Frec. Dither (Hz)** |
|---|---|---|---|---|
| 12A/8P (0x48) | 0,745 | 1356,852 | 0,136 | 1436,121 |
| 12A/8P (0x4C) | 0,765 | 1109,878 | 0,017 | 981,47 |
| 12A/8P (0x4E) | 1,47 | 778,21 | 0,02 | 24414,063 |
| 10 A/8P (0x45) | 3,792 | 422,833 | 0,009 | 24414,063 |
| 10 A/8P (0x4B) | 3,799 | 343,289 | 0,014 | 12207,031 |
| 10 A/8P (0x4F) | 3,772 | 269,833 | 0,01 | 12207,031 |
| 7 A/8P (0x44) | 0 | 0 | 0,042 | 12207,031 |
| 7 A/8P (0x48) | 0 | 0 | 0,014 | 97656,25 |
| 7 A/8P (0x4C) | 0 | 0 | 0,013 | 97656,25 |

Tabla 5-1: Datos de simulación RC

El rizado de conmutación es 3,4mV. En cuanto a los datos experimentales, la resolución del osciloscopio no es lo suficientemente grande como para poder medir los rizados tan pequeños que dará el ciclo límite, ya que dicha resolución es de 10mV por división frente a los pocos mV de amplitud de señal.

Como las oscilaciones de ciclo límite son un proceso aleatorio en la realidad, no podemos definir una frecuencia concreta de este fenómeno porque depende de la frecuencia de cambio de la variación de consigna que aplica la técnica del Dither y, en la realidad, la medida del ADC no será constante por factores de ruido del sistema o interferencias, por lo que, aunque sean variaciones muy ligeras, basta con modificar uno de los LSB para cambiar el resto que acumula el sumador asociado a la implementación del método, por lo que no habrá constancia en la frecuencia.

La siguiente tabla muestra los datos de simulación del convertidor Buck:

| **Caso (consigna)** | **Amp. LCO (mV)** | **Frec. LCO (Hz)** | **Amp. Dither (mV)** | **Frec. Dither (Hz)** |
|---|---|---|---|---|
| 12A/8P (0x45) | 17,63 | 1293,46 | 0,12 | 24414,063 |
| 12A/8P (0x48) | 34,19 | 971,704 | 0,25 | 12207,031 |
| 12A/8P (0x4D) | 10,41 | 3616,898 | 0,51 | 12207,031 |
| 10 A/8P (0x43) | 0 | 0 | 0 | 0 |
| 10 A/8P (0x49) | 12,72 | 4542,151 | 0,07 | 48828,125 |
| 10 A/8P (0x4E) | 43,73 | 1617,966 | 0,5 | 12207,031 |
| 7 A/8P (0x44) | 0 | 0 | 0,07 | 12207,031 |
| 7 A/8P (0x4A) | 0 | 0 | 0,12 | 48828,125 |
| 7 A/8P (0x4E) | 0 | 0 | 0,51 | 12207,031 |

Tabla 5-2: Datos de simulación Buck

El rizado de conmutación de simulación es 1,7mV. Se puede apreciar la atenuación que sufre el ciclo límite y su subida en frecuencia en los casos donde hay más resolución de ADC que de PWM. En cambio, en la situación de 7 bits de ADC y 8 de PWM, se genera un rizado de Dither pese a que, cuando no estaba aplicado, este era nulo.

Aunque realmente son medidas aproximadas debido a la aleatoriedad del LCO, la siguiente tabla mostrará las medidas correspondientes a los resultados experimentales del convertidor Buck:

| **Caso (consigna)** | **Amp. LCO (mV)** | **Amp. Dither (mV)** |
|---|---|---|
| 12A/8P (0x48) | 30 | 1 |
| 10 A/8P (0x43) | 20 | 0 |
| 10 A/8P (0x49) | 20 | 0 |
| 7 A/8P (0x44) | 0 | 0 |
| 7 A/8P (0x4A) | 0 | 0 |

Tabla 5-3: Datos experimentales Buck

# 6 Conclusiones y trabajo futuro

## *6.1 Conclusiones*

Los sistemas digitales son más utilizados gracias a ventajas que presentan frente a sus homólogos analógicos, como su resistencia al ruido, su fácil implementación y su reconfiguración. Pero no todo son bondades, ya que en los reguladores digitales en lazo cerrado pueden aparecer oscilaciones indeseadas a la salida, aunque esta esté estable en régimen permanente: estas son llamadas oscilaciones de ciclo límite. El problema que conllevan es que son señales que se acoplan a la salida y no poseen una amplitud o frecuencia fijas en la realidad. De hecho, refiriéndonos a la frecuencia, sabemos que son bajas, lo que dificulta su filtrado. Estas se producen cuando la resolución del ADC es mayor que la del PWM, por lo que el controlador podrá leer valores con más precisión de lo que puede actuar sobre la salida, haciendo que el error pueda no ser cero y oscile entre dos valores de consigna del PWM.

Por ello, en este Trabajo se ha implementado una técnica llamada Dither [Peterchev03], la cual consiste en variar la actuación a un valor próximo con el objetivo de tener una actuación intermedia, haciendo el error cero y reduciendo esta oscilación. Este método consigue subir en frecuencia y atenuar la amplitud de la señal indeseada (en el caso de que, tras implementarlo, siga habiendo algún tipo de oscilación), lo que hace más fácil al diseñador eliminarla con un filtro, ya que es más complicado eliminar las frecuencias bajas.

Su implementación es relativamente simple: primero se dividen los bits del ciclo de trabajo, los cuales antes iban conectados directamente al módulo PWM para que actuara, en un grupo que, nuevamente, serán de ciclo de trabajo y otro grupo de resto, que serán los LSB. Posteriormente, un acumulador tendrá la función de sumar en cada ciclo de conmutación los bits del resto y, en el caso de este Trabajo, cuando sume hasta llegar a un valor superior o igual a “16” (4 bits de resto) se incrementará el valor del ciclo de trabajo en “1”, lo que hará que la actuación oscile en poco tiempo, por lo que tendrá valores intermedios a los que tenía en el caso sin Dither, haciendo que el error pueda llegar a cero.

Por último, se podría decir que para evitar las oscilaciones del ciclo límite, el número de bits del ADC debe ser menor a los del PWM sumados con los del resto. En este Trabajo, el peor caso de ciclo límite corresponde con 12 bits de resolución del ADC frente a 8 del PWM, y es donde entra la técnica, porque, al aplicarla, conseguimos igualar la resolución del ADC con la del PWM (12 contra 8 de PWM + 4 del resto).

Los resultados experimentales realizados a lo largo de este Trabajo muestran que los efectos del ciclo límite se ven atenuados y, en algunos casos, eliminados completamente. Por el contrario, cuando se aplica la técnica en un sistema donde es imposible que haya oscilaciones no deseadas, aparecen acoplamientos debidos al propio Dither, aunque son de una amplitud muy pequeña y altas frecuencias.

## 6.2 Trabajo futuro

En cuanto al estudio del ciclo límite utilizando un filtro RC, se han obtenido resultados en simulación que afirman las condiciones de su aparición expuestas en el apartado 2.2.1. El problema es que, en las pruebas experimentales, la amplitud del rizado de la oscilación era tan pequeña que no se podía medir, por lo que sería necesario para continuar el estudio buscar otros filtros RC y otros reguladores para estos que hagan que suba notablemente la amplitud del ciclo límite, de tal forma que se pueda estudiar de forma experimental y no solo en simulación.

En el caso del convertidor Buck, se ha conseguido ver el LCO en las simulaciones y en las pruebas experimentales, aunque conviene que la amplitud de la oscilación indeseada sea mayor a la vista para poder continuar con el estudio y, así, contrastar resultados reales con y sin la mejora.

No todo queda en utilizar plantas con mejores características para ver el ciclo límite. También es necesario probar la técnica del Dither con distinto número de bits del resto: el resto que se ha utilizado en este Trabajo es de 4 bits (16 pasos), pero se debería probar desde 1 bit (se consiguen 2 pasos) hasta 6 bits (64 pasos) o más.

Existen otras soluciones a parte de la técnica del Dither para evitar las oscilaciones del ciclo límite cuando se tiene una resolución del ADC mayor a la del PWM, como ampliar la banda de tensiones que corresponden con el error nulo. Su implementación incluiría un regulador mejorado con una serie de instrucciones “if”, hablando en VHDL, con el objetivo de hacer que, aunque el sistema lea que el error no es cero, pero sí cercano, asuma que este es nulo, eliminando así el LCO.

# Referencias

[DeCastro03] A. De Castro Martín, “Aplicación del control digital basado en hardware específico para convertidores de potencia conmutados”. Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Industriales, 2003.

[DelBrio99] B. Martín Del Brío, “Sistemas Electrónicos basados en Microprocesasdores y Microcontroladores”. Revisión de Abril de 1999. Editorial Prensas Universitarias de Zaragoza (España)

[Erickson01] Robert W. Erickson, Dragan Maksimovic, “Fundamentals of Power Electronics”. University of Colorado. 2001.

[LinearADC] Linear Technology, “LTC1273/LTC1275/LTC1276. A/D converters with Reference”. Datasheet of LTC1273.

[Malik95] Norbert R. Malik, “Electronic Circuits. Analysis, Simulation and Design”. University of Iowa, Department of Electrical and Computer Engineering, 1995.

[Peterchev03] A.V. Peterchev, S.R. Sanders, “Quantization Resolution and Limit Cycling in Digitally Controlled PWM Converters”. IEEE Transactions on Power Electronics, Vol. 18, No. 1, Enero 2003.

[Sutter06] J-P. Deschamps, G.J.A. Bioul, G.D Sutter, “Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems”. John Wiley & Sons 2006.

[XilinxBoard] Xilinx, “Spartan-3 Starter Kit Board User Guide”. Datasheet of Spartan-3 Starter Kit Board.

[XilinxFPGA] Xilinx, “Spartan-3 FPGA Family Data Sheet”. Datasheet of Spartan-3 FPGA Family.

# Glosario

| | |
|---|---|
| ADC | Analog-to-Digital Converter (convertidor analógico-digital) |
| PWM | Pulse Width Modulation (modulador de ancho de pulso) |
| RC | Resistance-Capacitor (Resistencia-condensador) |
| DSP | Digital Signal Processor (procesador de señales digitales) |
| FPGA | Field-Programmable Gate Array (matriz de puertas programables) |
| MOSFET | Metal-Oxide-Semiconductor Field-Effect Transistor (transistor de efecto de campo) |
| LCO | Limit Cycling Oscillation (oscilación de ciclo límite) |
| DAC | Digital-to-Analog Converter (convertidor digital-analógico) |
| VGA | Video Graphics Array (matriz de gráficos de video) |
| DCM | Digital Clock Manager (gestor de reloj digital) |
| DIP | Dual In-Line (encapsulado de doble línea de pines) |
| LSB | Less Significant Bit (bit menos significativo) |

# Anexos

## A. Regulador del filtro RC de rápida actuación

El primer diseño que se hizo no valió para apreciar el efecto del ciclo límite, pero se pudo saber más sobre él.

Este regulador tiene como ecuación en diferencias $\frac{z-0,99}{z-1}$ y su comportamiento final es el que muestra la siguiente figura.

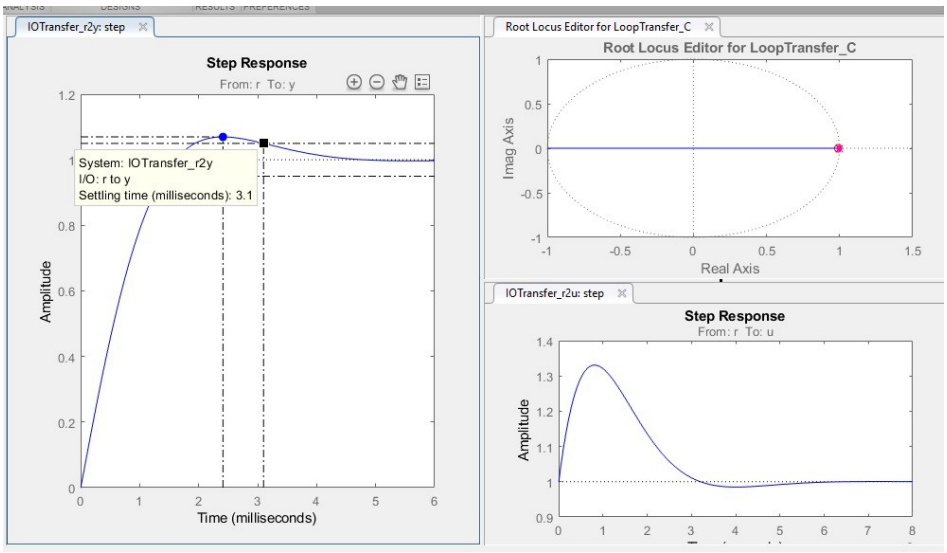


Figura A-1: Tiempo de establecimiento del regulador RC rápido

Se puede observar que la actuación sobreoscila y su tiempo de establecimiento es de 3,1 ms.

El problema se descubrió al ver el tiempo de establecimiento de la planta. Como los tiempos de establecimiento de la planta y del regulador son tan parecidos, a la planta no le da tiempo a actuar sobre la señal de salida, por lo que, aunque se tienen más bits de ADC que de PWM, el regulador va a ejercer un efecto de "autodither" sobre la actuación y, consecuentemente, no se verá ciclo límite. A continuación, se muestra una figura perteneciente a la simulación del modelo del filtro RC con un ADC de 12 bits y PWM de 8 bits sin la aplicación de Dither.

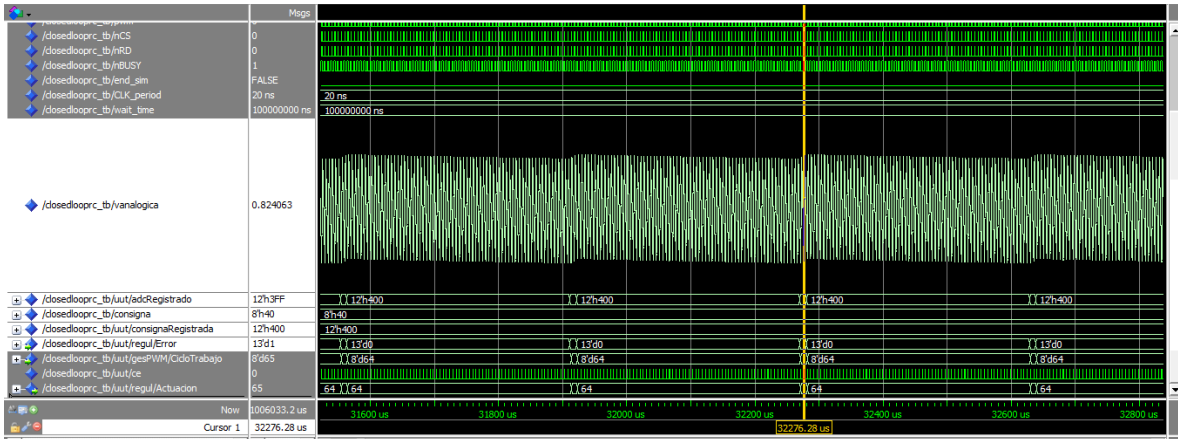


Figura A-2: Simulación del regulador RC rápido

Es destacable cómo cambia la actuación periódicamente de 64 a 65, como si fuera un módulo PWM con la mejora del Dither implementada, pero en este caso no lo está. Se aprecia que existe un pequeño rizado acoplado al de conmutación. Se trata del rizado generado por el "autodither", que en este caso tiene una amplitud de 1,04 mV y el de conmutación de 3.4 mV.

## B. *Medidas con 10 bits de resolución de ADC en RC*

A continuación, se mostrarán capturas de las simulaciones tanto de los casos con Dither como los que no lo tienen con el objetivo de ver los efectos tan diversos que se aprecian al introducir una consigna distinta en el sistema:

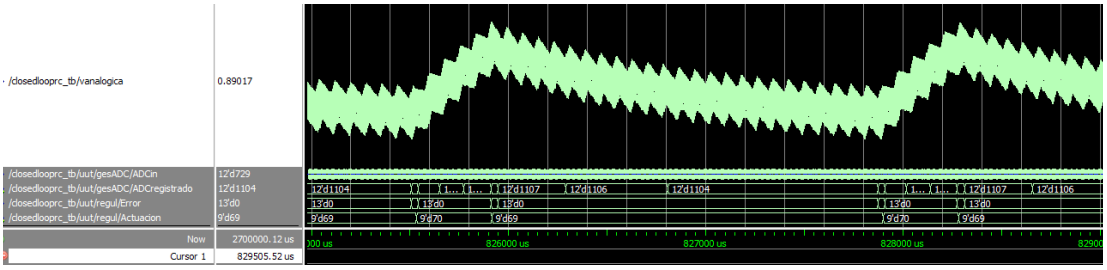


Figura A-3: Simulación de 10 bits de ADC sin Dither RC (periodo)

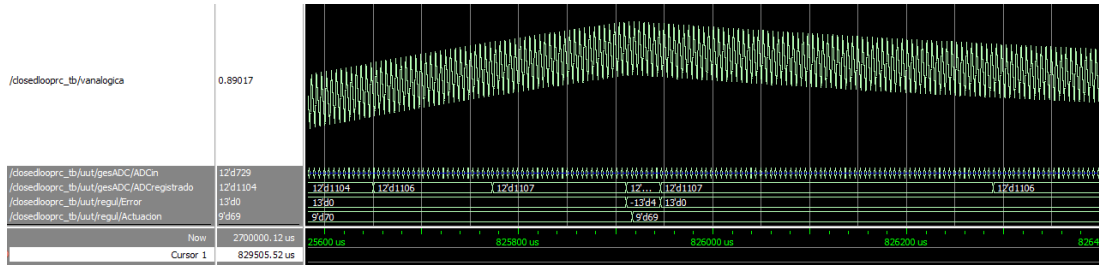


Figura A-4: Simulación de 10 bits de ADC sin Dither RC (amplitud)

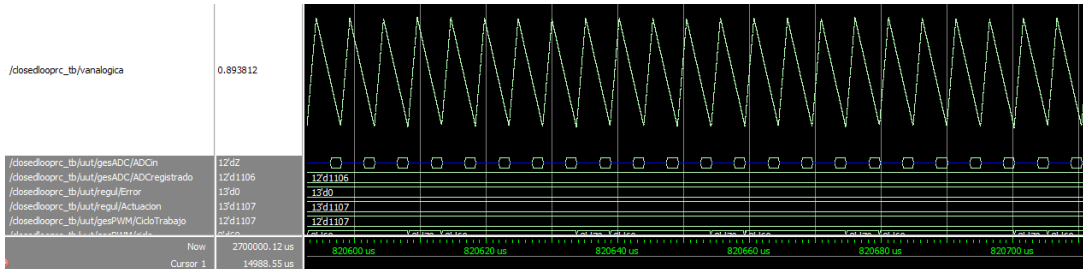


Figura A-5: Simulación de 10 bits de ADC con Dither RC

Se puede apreciar que, para este caso (distinto del ya visto en el apartado 5.2.1), se vuelve a eliminar el ciclo límite.

En la prueba experimental, se volverá a ver solamente el rizado de conmutación debido a la resolución del osciloscopio, tal y como se muestra a continuación:

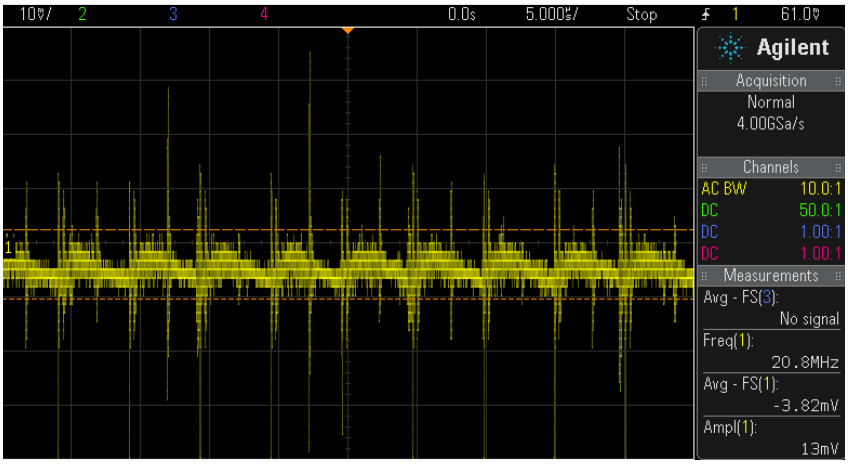


Figura A-6: Prueba experimental de 10 bits de ADC sin Dither RC

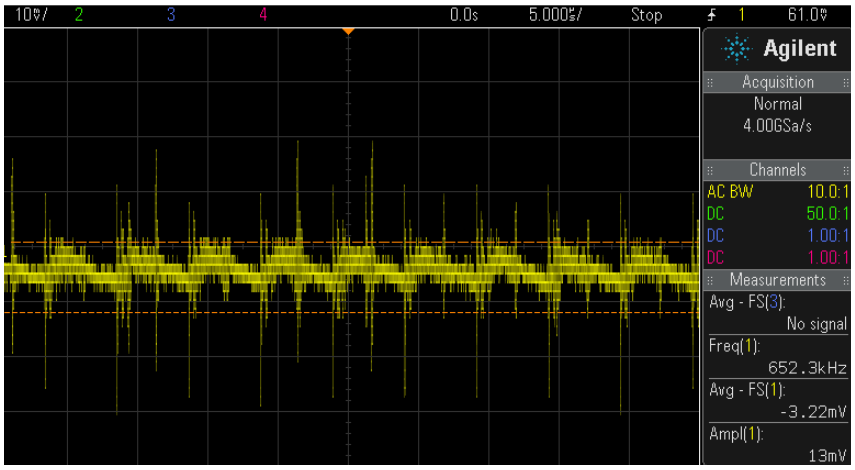


Figura A-7: Prueba experimental de 10 bits de ADC con Dither RC

## C. *Medidas con 12 bits de resolución de ADC en RC*

Para otras consignas se han obtenido los siguientes resultados:

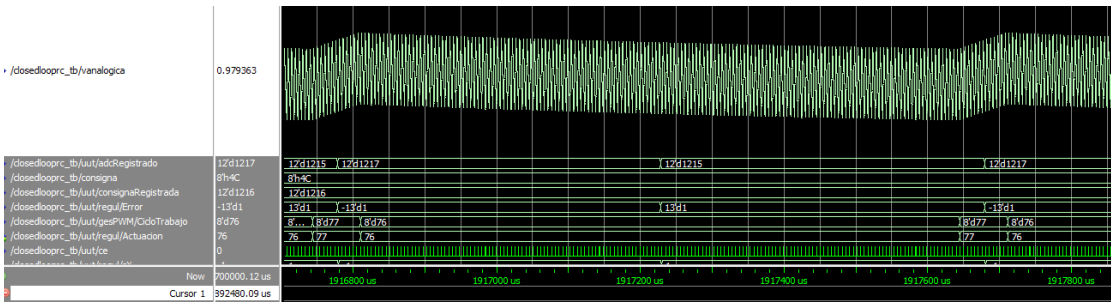


Figura A-8: Simulación de 12 bits de ADC sin Dither RC

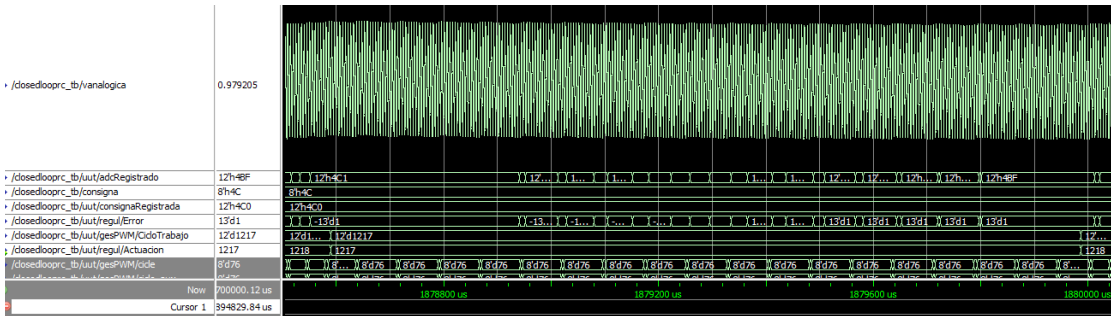


Figura A-9: Simulación de 12 bits de ADC con Dither RC

Se puede apreciar que el ciclo límite no desaparece completamente, pero se ve atenuado en gran medida.

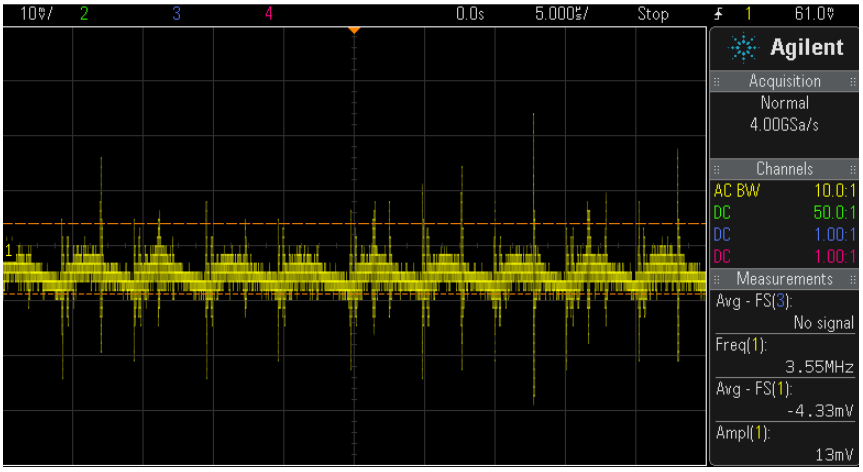


Figura A-10: Prueba experimental de 12 bits de ADC sin Dither RC

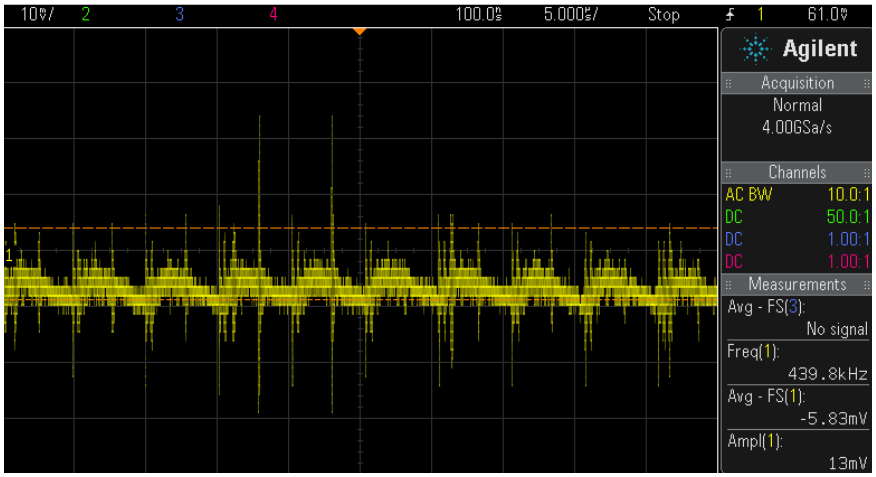


Figura A-11: Prueba experimental de 12 bits de ADC con Dither RC

Para terminar este apartado, se mostrará el tiempo de establecimiento medido experimentalmente para el sistema del filtro RC (es común para los otros casos de resolución de ADC).

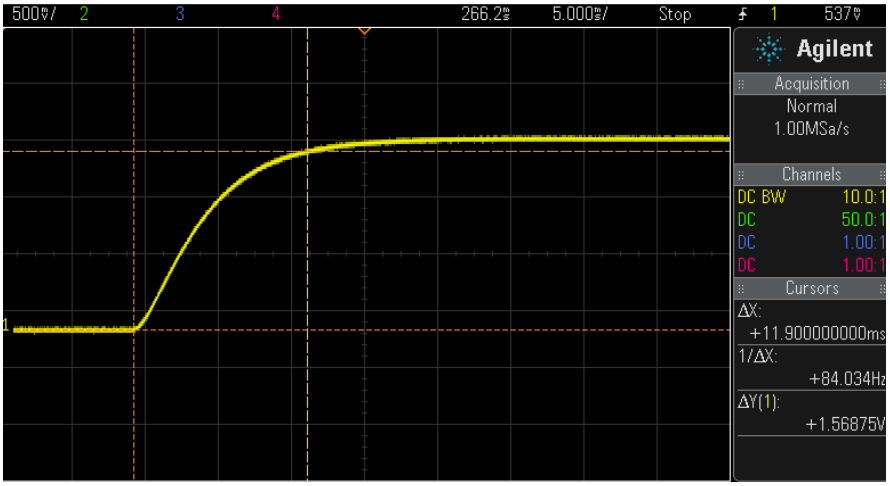


Figura A-12: Tiempo de establecimiento del sistema RC

## D. *Medidas con 10 bits de resolución de ADC en Buck*

Igual que se ha hecho con los sistemas del filtro RC, se hará lo propio con los del Buck.

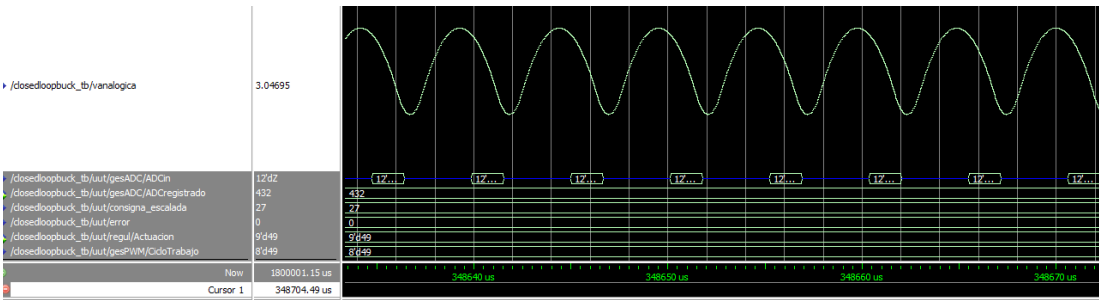


Figura A-13: Simulación de 10 bits de ADC sin Dither Buck (1)

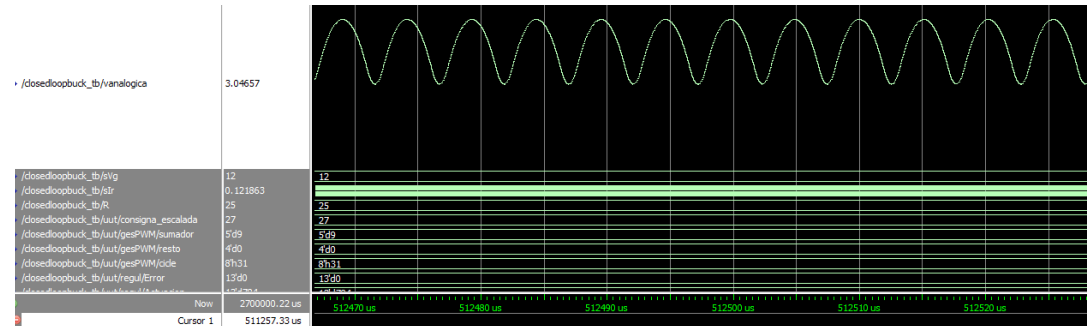


Figura A-14: Simulación de 10 bits de ADC con Dither Buck (1)

En este primer caso, se puede ver que no hay ciclo límite cuando no hay Dither porque la actuación se mantiene constante y, en su simulación con Dither, tampoco aparece el rizado de Dither.

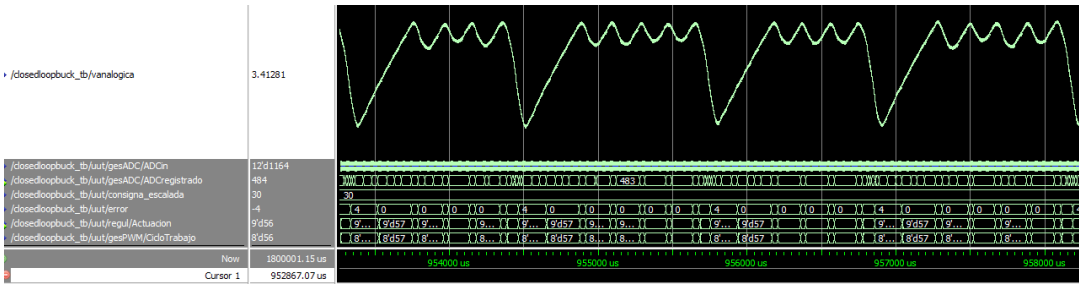


Figura A-15: Simulación de 10 bits de ADC sin Dither Buck (2)

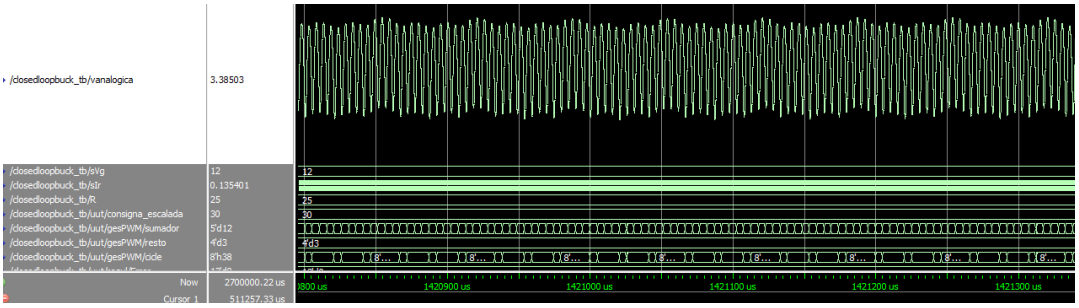


Figura A-16: Simulación de 10 bits de ADC con Dither Buck (2)

En cambio, en este otro caso se aprecia una oscilación muy grande comparada con el rizado de conmutación que, tras aplicar el Dither, se verá atenuada en gran medida.

Midiendo experimentalmente, se obtiene lo siguiente:

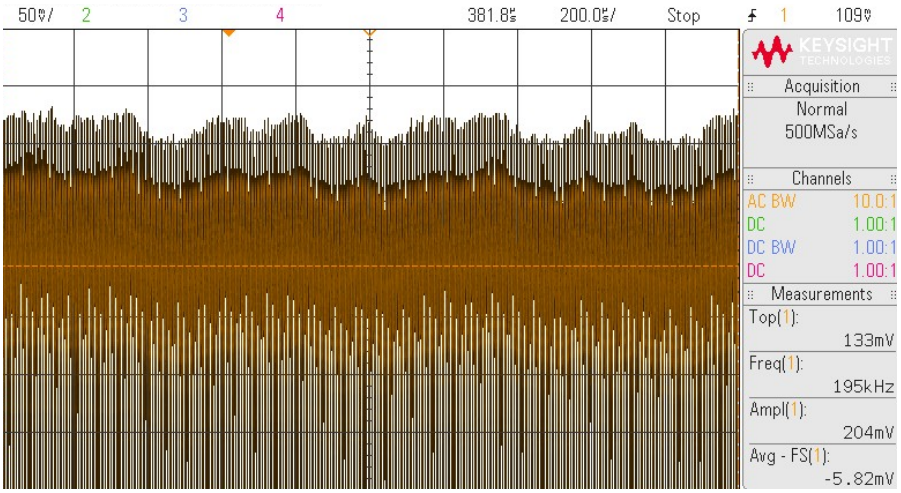


Figura A-17: Prueba experimental de 10 bits de ADC sin Dither Buck

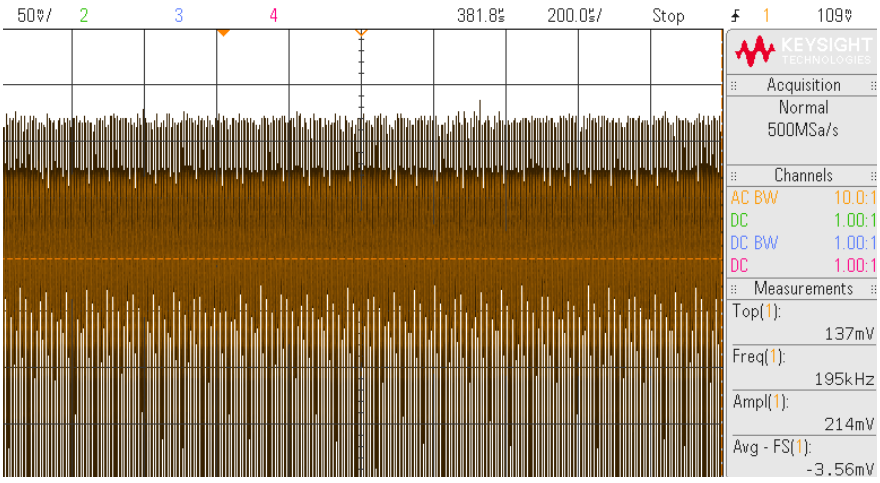


Figura A-18: Prueba experimental de 10 bits de ADC con Dither Buck

## E. *Medidas con 12 bits de resolución de ADC en Buck*

Ahora se verán más resultados obtenidos para el caso de 12 bits de ADC con el convertidor Buck:

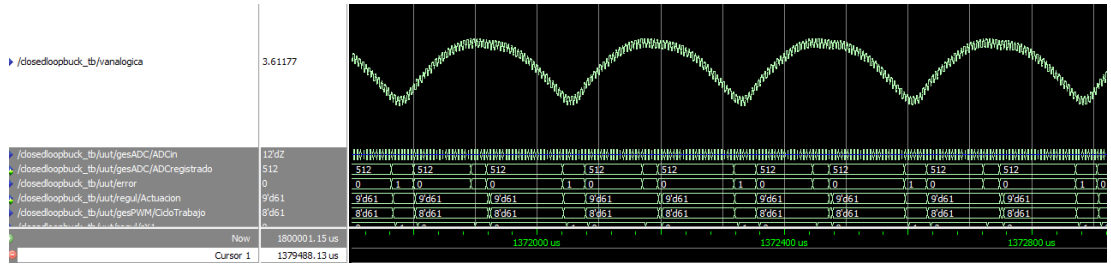


Figura A-19: Simulación de 12 bits de ADC sin Dither Buck

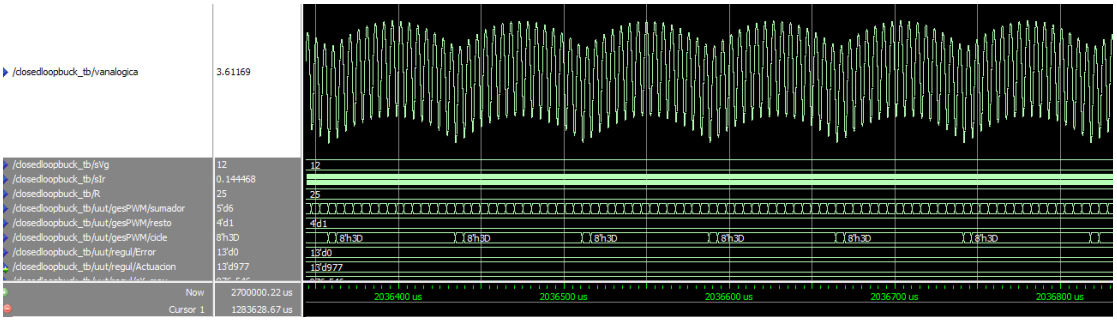


Figura A-20: Simulación de 12 bits de ADC con Dither Buck

Nuevamente, se puede ver como se atenúa la oscilación indeseada y como sube en frecuencia.

En cambio, para esta consigna en una prueba experimental se puede observar cómo coincide que no hay ciclo límite, pese a la gran diferencia de bits entre ADC y PWM:

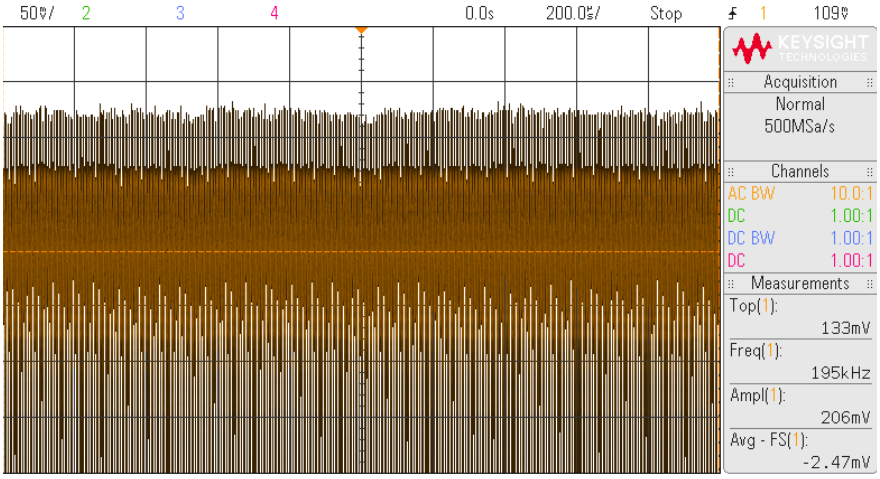


Figura A-21: Prueba experimental de 12 bits de ADC sin Dither Buck

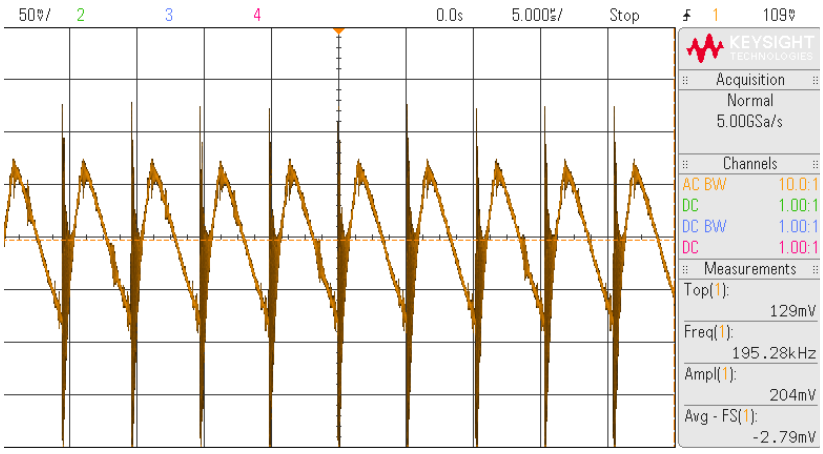


Figura A-22: Prueba experimental de 12 bits de ADC con Dither Buck

Para concluir este apartado, a continuación, se mostrará una gráfica con el tiempo de establecimiento para el sistema del convertidor Buck, siendo este común para los otros casos de ADC.

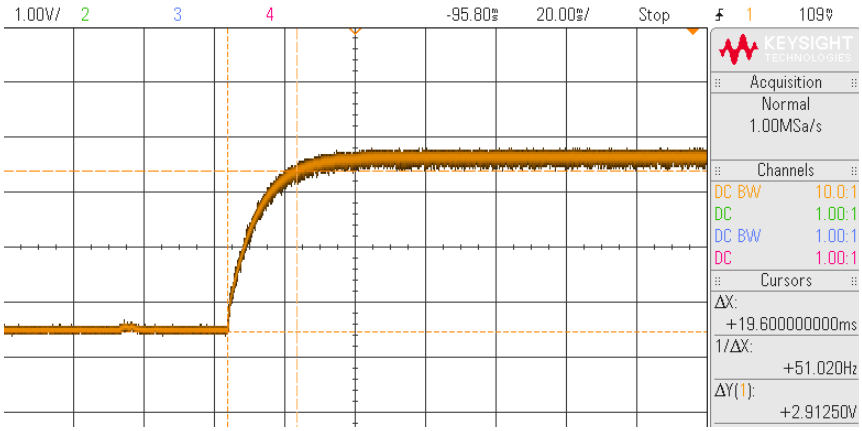


Figura A-23: Tiempo de establecimiento del sistema Buck